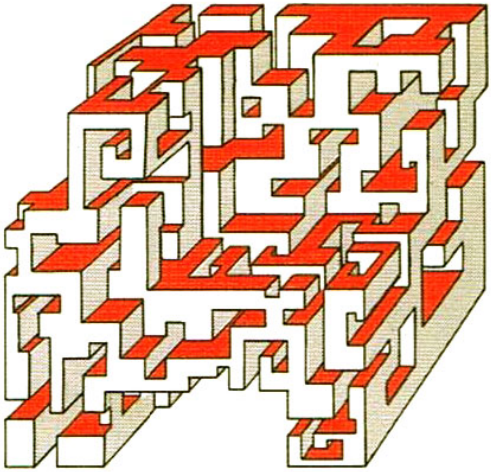


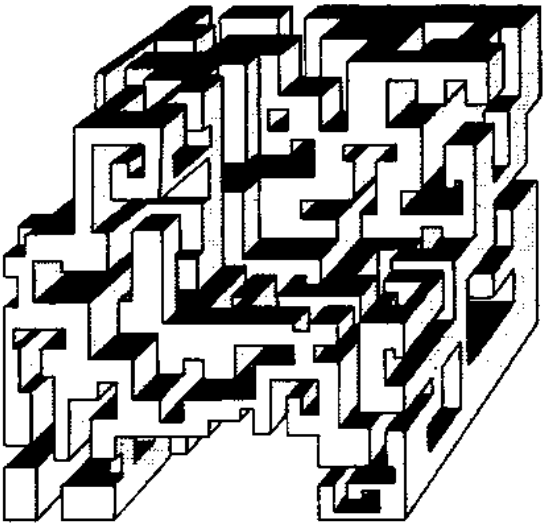
Voß

Einführung in die

Künstliche Intelligenz

Mit vielen Programmen für C 64

EIN DATA BECKER BUCH



Voß

Einführung in die Künstliche Intelligenz

Mit vielen Programmen für C 64

EIN DATA BECKER BUCH

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

G L I E D E R U N G

Seite

Vorwort 1

E r s t e r T e i l :

Theoretische Hintergründe 7

Kap. 1 : Grundbegriffe 9

1.1 Vorbemerkung 9
1.2 Intelligenz 10
1.3 Leben 18
1.4 Werkzeuge 27
1.5 Denken 36
1.6 Künstliche Intelligenz ... 46

Kap. 2 : Ein Blick in die Vergangenheit 53

2.1 Vorbemerkung 53
2.2 Die Anfänge 54
2.3 Mechanische Rechner 57
2.4 Von Hollerith bis Zuse ... 59
2.5 "Elektronengehirne" 61
2.6 Mikroprozessor-Technik ... 63
2.7 Die Zukunft 64

Kap. 3 : Anwendungsbereiche der künstlichen Intelligenz .. 67

3.1 Vorbemerkung 67
3.2 Computer und andere Maschinen 68
3.3 "Denkmaschinen" 72

Zweiter Teil :

Der Computer 85

Kap. 4 : Zur Funktionsweise moderner Rechner 87

4.1 Vorbemerkung 87
4.2 Grundbegriffe 88
4.3 Datenverarbeitung 90
4.4 Codes 92
4.5 Hardware 94

Kap. 5 : BASIC - Grundlagen 97

5.1 Vorbemerkung 97
5.2 Ergebnisausgabe 98
5.3 Wertzuweisungen 101
5.4 Informationseingabe 103
5.5 Programmverzweigungen 105
5.6 Stringbearbeitung 109
5.7 Graphik 110

D r i t t e r T e i l :

Programme für den C 64 115

Kap. 6 : Einführung 117

6.1 Zielsetzungen 117
6.2 Problemanalyse 119
6.3 Die Demonstrationsbeispiele 132

Kap. 7 : Expertensystem 137

Kap. 8 : Auskunftssystem 173

Kap. 9 : Suchprogramm 191

Kap. 10 : Entscheiden 213

Kap. 11 : Erkennen 237

Kap. 12 : Taktisches Spiel 253

Kap. 13 : Selbstlernende Programme . 287

Kap. 14 : Kunst 311

Kap. 15 : Dialog 325

Kap. 16 : Reaktion 341

Kap. 17 : Unterricht 355

Kap. 18 : Weitere Programme 373

Literaturhinweise

Stichwortverzeichnis

Vorwort

Seitdem die ersten Großcomputer, die unglaublich teuren Giganten der fünfziger Jahre, uns zeigten, wie auch kompliziertere Rechnungen in Sekundenbruchteilen durchgeführt werden - und dazu noch fehlerfrei - ist bekannt, daß zur Durchführung solcher Aufgaben Programme benötigt werden. Dies gilt bis heute. Auch die modernen Kleinrechner, die so preiswert geworden sind, daß sie schon in vielen Haushalten stehen - und übrigens mehr leisten können als die sog. "Elektronengehirne" der fünfziger Jahre - auch sie werden über Programme gesteuert.

Programme geben dem Computer die Anweisungen, die er benötigt, um ein arithmetisches Mittel zu berechnen, um Datenbestände zu sortieren, um Texte zu korrigieren, um Bilder zu zeichnen, um Abenteuerspiele zu präsentieren, um physikalische Abläufe zu simulieren usw.

Diese Programme muß der Computerbenutzer selbst erstellen (oder er muß sie sich kaufen; dann sind sie von anderen erstellt worden), d.h. der Rechner tut nichts, ohne ein vom Benutzer vorgegebenes Programm.

Mit wachsender Leistungsfähigkeit der Rechner, insbesondere aufgrund der stark wachsenden Speicherkapazitäten, ist es nun möglich geworden, fertige Programme schnell abrufbar bereitzustellen. Damit wird erreicht, daß der Computerbenutzer mehr und mehr von der Last eigener Programmerstellung befreit wird : Der Computer erledigt - quasi per Knopfdruck - von selbst bestimmte Arbeiten, ohne daß der Computerbenutzer die entsprechenden Programme erstellen müßte. Der Computer kann längere Zeit ohne zusätzliche Informationen auskommen, er

arbeitet gewissermaßen selbständig und wird dies in Zukunft um so mehr tun, je weiter die Speichermöglichkeiten verbessert werden.

Mit der auch zukünftig zu erwartenden Erhöhung der Speicherkapazitäten wächst die "Selbständigkeit" der Computer - zumindest so weit ihre konkrete Nutzung durch den Benutzer betroffen ist.

Hinzu kommt, daß in Zukunft Computer zu erwarten sind, die mit völlig neuen Speichertechnologien und mit veränderten, darauf abgestellten Rechnerarchitekturen arbeiten werden und dabei die Selbständigkeit von Prozessen, Abläufen, Berechnungen usw. weiter erhöhen werden. In den großen Forschungslabors, insbesondere in den USA und in Japan, arbeitet man an diesen Entwicklungen (Biochips, Informationsspeicherung auf molekularer Ebene usw.) und bereitet so die 5. Generation von Computern vor.

Nicht nur wegen der zunehmenden Selbständigkeit der Computer, sondern auch wegen der vorstellbaren zukünftigen Nähe der kommenden computerinternen Strukturen (Eiweißmoleküle als Speichermedien) zur "organischen Informationsverarbeitung", drängt sich die Frage auf, ob und wie weit Computer in der Lage sind, intelligente Arbeiten zu übernehmen, bzw. Tätigkeiten des menschlichen Gehirns auszuführen oder, was vielleicht einfacher ist, solche Tätigkeiten zu simulieren.

Werden Computer intelligent ?
Sind sie vielleicht schon intelligent ?
Verfügen sie über künstliche Intelligenz ?

Dies sind Fragen, die immer mehr interessieren - allein schon deshalb, weil mehr und mehr Personen sich mit Computern befassen.

Allerdings sind diese Fragestellungen keineswegs neu : In dem Moment, als die ersten Rechner funktionierten, in dem Moment also, als Geräte zur Verfügung standen, die geistige Tätigkeiten übernehmen oder nachvollziehen konnten, da nannte

man diese Riesengeräte "Elektronengehirne". In dieser Bezeichnung kam die Vorstellung oder vielleicht auch die Angst zum Ausdruck, daß plötzlich Geräte existierten, die Aufgaben erledigen konnten, für die bisher offenbar nur das menschliche Gehirn zuständig war.

Seit dieser Zeit ist die Frage aktuell, was denn überhaupt "künstliche Intelligenz" sei - und seit dieser Zeit werden auf diese Frage die unterschiedlichsten Antworten gegeben :
Die einen gehen so weit, daß sie nicht nur die Intelligenz von Computern behaupten, sondern ernsthaft diskutieren, ob Computer leben (SIMONS,G.: 1984), andere sprechen nach wie vor von den "Blechkisten", denen notwendigerweise jeder Funke von Intelligenz fehlen muß.

Dieses Buch setzt sich die Aufgabe, diese Frage zu versachlichen und vorurteilsfrei zu diskutieren. Dabei geht es insbesondere darum, die zentralen Begriffe, wie zum Beispiel Intelligenz, Leben, Denken, unter dem Gesichtspunkt zu betrachten, welche Bedeutung ihnen beim Rechnereinsatz zukommt.

Diese zentralen Begriffe werden durch Rückgriff auf die zu diesen Themen vorliegende Literatur erörtert. Damit werden dem Leser eine große Zahl unterschiedlicher Argumente vorgestellt, die es ihm dann erlauben, sich selbst ein Urteil über den Begriff der künstlichen Intelligenz zu bilden. Es wird dann auch beurteilen können, welche Bedeutung diesem Begriff beim Einsatz von Kleinrechnern zukommt.

Die Kleinrechner, vor allem die sog. Home-Computer verbreiten sich ja derzeit so rasch, daß eine sehr schnell anwachsende Zahl von Computerbenutzern sich sehr ernsthaft mit diesen Begriffen zu beschäftigen beginnt.

Deshalb beziehen sich die Erörterungen dieses Buchs und die dargestellten Programmbeispiele auf den derzeit in der Bundesrepublik Deutschland

meistgenutzten Kleinrechner, nämlich den

Commodore C64

Gleichwohl gilt natürlich, daß die generellen Ausführungen dieses Buchs auch für denjenigen Leser von Interesse sind, der einen anderen Rechner oder gar keinen besitzt. Die Beispiele wie auch die theoretischen Erörterungen sind so angelegt, daß praktisch keine Vorkenntnisse vorausgesetzt werden.

Auch derjenige, der sich zum ersten Mal mit dem Thema "Künstliche Intelligenz" beschäftigt, soll dieses Buch lesen und die Programmbeispiele nachvollziehen können.

Wir wollen damit erreichen, daß der Besitzer eines Kleinrechners sich nicht darauf beschränkt, Weltraumschlachten und andere Computerspiele durchzuprobieren, und auch nicht darauf, eigene Programme zu entwickeln ,auch wenn dies eine sehr wichtige Angelegenheit ist, sondern er soll mit diesem Buch dazu geführt werden, besser die Bedeutung von Computern einzuschätzen.

In Zukunft wird es in erster Linie nämlich nicht darauf ankommen, welche neuen Spiele man mit dem Computer spielen kann, oder welche Rechenaufgaben er mit welchen Programmen durchrechnet, sondern darauf, welcher Art die Beziehungen des Informationsaustauschs zwischen Mensch und Computer sind, wie die Struktur und der Typus der Kommunikationsbeziehungen sich darstellen und wie deshalb Problemlösungswege prinzipiell auszusehen haben.

Die derzeit zur Verfügung stehenden Rechner und die in der weiteren Zukunft einsetzbaren Computer werden ihren Nutzen nämlich nur dann voll entfalten können, wenn diejenigen, die privat oder beruflich damit umzugehen haben, wissen, auf welchen Ebenen, unter welchen Voraussetzungen und

mit welchen Chancen die Kommunikation zwischen Nutzer und Computer - angesichts der zunehmenden Leistungsfähigkeit dieser Geräte - sich vollziehen kann.

Der Umgang mit diesen Kommunikationsmöglichkeiten und ihre Beurteilung sind für die Bewältigung der Zukunftsgefahren und für die Nutzung der zukünftigen Chancen - soweit der Computereinsatz betroffen ist - außerordentlich wichtig und bezeichnen somit das Hauptanliegen dieses Buches. Um diesem Anliegen gerecht zu werden, gehen wir im einzelnen folgendermaßen vor :

Der erste Teil ist eher theoretischer Natur und beschäftigt sich insbesondere mit den im Rahmen dieser Themenstellung relevanten Grundbegriffen. Hier wird beispielsweise geklärt, was unter "Künstlicher Intelligenz" überhaupt verstanden werden soll.

Im zweiten Teil befassen wir uns - soweit dies für die Zwecke der dann folgenden Ausführungen erforderlich ist - mit der Funktionsweise von Rechnern und wir besprechen auch diejenigen Elemente der Programmiersprache BASIC, die für die Entwicklung der Demonstrationsbeispiele benötigt werden.

Im dritten Teil - dem Hauptteil - werden dann BASIC-Programmbeispiele vorgeführt, die illustrativ erläutern sollen, welche verschiedenen Anwendungsbereiche mit dem Stichwort "Künstliche Intelligenz" angesprochen werden. Natürlich kann es sich dabei nur um recht kleine und bescheidene Illustrationsbeispiele handeln, weil "echte" Programme sofort den Rahmen dieser Einführung sprengen würden.

Hauptanliegen dieser Beispiele ist es, die Grundprinzipien solcher Anwendungsprobleme zu verdeutlichen. Der Leser, der diese Prinzipien verstanden hat, ist dann in der Lage, selbst größere und praktisch nutzbare Programme zu entwerfen.

Die einzelnen Programmbeispiele stehen stellvertretend für generelle Einsatzgebiete der sog. KI-Programme, weshalb einleitend jeweils das angesprochene Problemgebiet kurz umrissen wird. Es folgt dann eine ausführliche Analyse desjenigen Problems, das als Illustrationsbeispiel ausgewählt wurde. In einem weiteren Arbeitsschritt stellen wir dann das Programm selbst vor, das durch die Angabe der Variablenliste und durch eine schrittweise Programmbeschreibung dokumentiert wird. Schließlich werden auch die Programmergebnisse vorgestellt.

Im einzelnen ist jeder Abschnitt des Hauptteils folgendermaßen aufgebaut :

- Beschreibung des Problembereichs
- Skizzierung der Lösungsansätze
- Vorstellung des Beispiels
- Problemanalyse zum Beispiel
- BASIC-Programm
- Variablenliste
- Programmbeschreibung
- Programmergebnisse
- Ausblick (Ergänzungen, Alternativen)

Abschließend sei auf das Literaturverzeichnis verwiesen, wo der Leser zu den einzelnen Kapiteln dieses Buchs weiterführende Literatur finden kann, die ihm zum Weiterstudium und zur Vertiefung der hier angesprochenen Themenbereiche empfohlen sei.

ERSTER TEIL:

Theoretische Hintergründe

Kapitel 1 : Grundbegriffe

1.1 Vorbemerkung

Der Hauptteil dieses Buches besteht darin, Programme für den Commodore C64 vorzustellen, die als kleine Illustrationsbeispiele der "Künstlichen Intelligenz" interpretiert werden können. Es ist jedoch nicht sinnvoll, sofort mit derartigen Beispielen quasi "mit der Tür ins Haus" zu fallen. Der Leser sollte sich zunächst mit den zentralen Grundbegriffen vertraut machen.

Nach der Diskussion dieser Grundbegriffe ist er in der Lage, zu einem eigenen Urteil über die Einsatzmöglichkeiten und -grenzen von Computern zu gelangen und er kann dann vielleicht abschätzen, was es mit dem Begriff der "Denk"-Maschinen überhaupt auf sich hat. Es ist an der Zeit, den interessierten Homecomputer-Benutzer nicht nur mit Programmen, die er abtippen kann, oder mit Problemlösungen, die ihm vielleicht bislang gefehlt haben, zu versorgen, sondern es sollen auch Hintergrundinformationen bereitgestellt werden.

Das Verständnis für derartige Informationen hängt in entscheidender Weise davon ab, daß mit einem klar begrenzten und definierten begrifflichen Instrumentarium gearbeitet wird. Deshalb also wird den weiteren Ausführungen dieses Kapitel zur Klärung der verwendeten Begriffe vorangestellt.

1.2 Intelligenz

Ganz ohne Zweifel ist der entscheidende Begriff, um den es in diesem Zusammenhang geht, der Begriff der Intelligenz : Die Diskussion der Frage, ob es intelligente Maschinen gibt oder eines Tages geben könnte, Maschinen also, die mit "künstlicher" Intelligenz ausgestattet sind, erfordert es, an erster Stelle diesen Begriff näher zu beleuchten.

Mit dieser ersten Begriffsklärung geraten wir schon in große Probleme. Es gibt nämlich nicht nur eine und damit vielleicht verbindliche Definition des Begriffs der Intelligenz, sondern es gibt sehr viele durchaus unterschiedliche Definitionen. Einige davon wollen wir kurz vorstellen, um danach zu überprüfen, ob sie es ermöglichen, zu einer präziseren Vorstellung darüber zu gelangen, was unter "Intelligenz" verstanden werden könnte.

In einem normalen Lexikon finden wir folgende Begriffsklärung :

" Intelligenz (lat.), komplexe Fähigkeit zu Leistungen, die durch spontanes Erfassen von Zusammenhängen in neuen Situationen erzielt werden. I. findet sich schon bei höheren Tieren; für den Menschen ist sie zur Lebensbewältigung unentbehrl. und hat darüber hinaus entscheidenden Anteil an der Kulturentwicklung." (FISCHER - LEXIKON, Fischer Taschenbuch Verlag, 1975, Bd.9,S.2843).

In dieser Definition tauchen einige Begriffe auf, die ihrerseits eigentlich einer Definition bedürfen :

Es ist die Rede von einer "komplexen Fähigkeit", von "spontanem Erfassen", von "Lebensbewältigung" und von "Kulturentwicklung".

Generell dürfen wir davon ausgehen, daß alle Definitionen, die ihrerseits definitionswürdige Begriffe enthalten, "Verlegenheitslösungen" sind : Man versucht, den zu definierenden Sachverhalt möglichst geschickt zu umschreiben, und auf die bei dieser Umschreibung verwendeten Begriffe geht man dann lieber nicht näher ein.

Trotz dieser Kritik an der obigen Definition sollten wir das Stichwort "Lebensbewältigung" im Gedächtnis behalten :

Verfügte der Mensch nicht über das, was wir Intelligenz nennen, so könnte er sein Leben nicht bewältigen. Er würde mit recht hoher Wahrscheinlichkeit in unvorhergesehenen oder schwierigen Situationen scheitern und die Gattung "Mensch" wäre schon längst ausgestorben, wäre der Mensch nicht intelligent.

Intelligenz erklärt sich also in diesem Zusammenhang aus ihrem Ergebnis heraus, nämlich aus dem Überleben der Gattung "Mensch" unter sich verändernden Umweltbedingungen und auch aus der Art und Weise, wie diese Gattung ihre Lebensbedingungen und -verhältnisse selbst gestaltet, verändert und verbessert.

In ähnlicher Weise, wie es in der obigen Definition zum Ausdruck kommt, äußert sich der Psychologe P.R.HOFSTÄTTER :

" Intelligenz. Die in sehr großer Zahl vorliegenden Definitionen der Intelligenz betonen im wesentlichen vier Sachverhalte : 1. daß es sich um eine Begabung bzw. um eine Gruppe von Begabungen handelt, die ein Lebewesen in höherem oder geringerem Maße besitzen kann; 2. daß diese Fähigkeit die Lösung konkreter oder abstrakter Probleme und damit die Bewältigung neuartiger Situationen ermöglicht; 3. daß sie das bloße Herumprobieren und das Lernen an dessen sich zufällig einstellenden Erfolgen weitgehend erübrigt; 4. daß diese Begabung sich in der

Erfassung, Anwendung, Deutung und Herstellung von Beziehungen und Sinnzusammenhängen äußert." (HOFSTÄTTER,P.R.: Intelligenz, 1957, S.172).

Weiter führt er aus, daß von der Intelligenz in weitem Maße (unter einigermaßen normalen Verhältnissen) der "Lebenserfolg des Individuums" abhängt (ebenda, S.173). Diese Aussage korrespondiert genau mit dem, was weiter oben diskutiert wurde.

Insoweit ermöglicht also Intelligenz ein zielgerichtetes Verhalten , wobei das generelle Ziel das der Lebensbewältigung ist. Diesem Generalziel ordnen sich viele und durchaus unterschiedliche Teilziele unter : Vom erfolgreichen Versuch, sich irgendwelcher Feinde zu erwehren, über das ungefährdete Überqueren einer viel befahrenen Straße, bis hin zur erfolgreichen Zubereitung einer Portion Bratkartoffeln.

Wir dürfen aber nicht übersehen, daß bei der Definition mit Hilfe des Begriffs des zielgerichteten Verhaltens auch Probleme der Interpretation auftauchen :

Ist der Raubmörder, der (sehr zielgerichtet) einen einsamen Wanderer erschlägt, um ihn seiner Barschaft zu berauben, deshalb, weil er sehr konsequent sein Ziel verfolgt, intelligent ? Ist er intelligenter als sein Opfer, welches in Sekundenbruchteilen Dutzende von alternativen Rettungsmöglichkeiten in seinem Kopf (sicherlich unter Benutzung von dem, was wir Intelligenz nennen wollen) kalkuliert, mit diesen Kalkulationen aber nicht fertig wird, bevor ein Knüppelhieb sein Leben beendet ? Im Sinne der Lebensbewältigung war dieses vielleicht hochintelligente Opfer dem vielleicht schwachsinnigen Raubmörder ganz offenbar deutlich unterlegen.

Diese kleine Überlegung am Rande soll verdeutlichen, daß die hier notwendige Klärung des

Intelligenzbegriffs doch auf beträchtliche Schwierigkeiten stößt. Häufig wird deshalb die Ansicht vertreten, daß eine Definition, die sich an dem Ziel der Lebensbewältigung orientiert, möglicherweise zu kurz greift, und daß genau deshalb viel zu unüberlegt oder zumindest zu rasch von "Künstlicher Intelligenz" gesprochen wird.

Wenn "Künstliche Intelligenz" bedeuten sollte, daß die Konstruktion einer Maschine "nach dem Bilde des Menschen" möglich sein sollte, "eines Roboters, der seine eigene Kindheit haben, Sprachen wie ein Kind lernen und sein Wissen von der Welt dadurch erlangen soll, daß er die Welt durch seine eigenen Sinnesorgane erfährt..." (WEIZENBAUM,J. : 1978, S.268), dann darf (so WEIZENBAUM) der Begriff der Intelligenz nicht zu eng definiert werden :

"Ein Organismus (wird) weitgehend durch die Probleme definiert, denen er sich gegenübersieht. Der Mensch muß Probleme bewältigen, mit denen sich keine Maschine je auseinandersetzen muß, die von Menschenhand gebaut wurde." (ebenda, S.269).

WEIZENBAUM bringt mit dieser Aussage zum Ausdruck, daß zwar auch der Mensch ein Wesen ist, das Informationen verarbeitet (insoweit also einem Computer vergleichbar), daß die menschliche Intelligenz sich aber, wegen der prinzipiellen Nichtvergleichbarkeit der Probleme, denen sich der Mensch im Gegensatz zum Computer gegenübersieht, von jeder "Künstlichen Intelligenz" (sollte es sie überhaupt geben) fundamental unterscheiden muß.

Diese Überlegungen, wie auch schon das Beispiel von dem Raubmörder, zeigen, daß der Begriff der Intelligenz nicht allein an der Zielgerichtetheit des Handelns orientiert werden kann. Zusätzliche Komponenten müssen hinzutreten, wie aus den folgenden Ausführungen deutlich wird :

In einem einführenden Lehrbuch der Psychologie wird betont, daß einzelne Forscher sehr wohl unterschiedlicher Ansicht darüber sind, was

Intelligenz sei (BÜHLER,C. : 1962, S.99). Hervorzuheben ist "die Fähigkeit, neue Aufgaben zu lösen, neue Probleme zu meistern."(ebenda). Andere Forscher hingegen betonen die Lernfähigkeit als das entscheidende Merkmal der Intelligenz, d.h. die Fähigkeit, bisherige Erfahrungen zu verwerten.

"Außer diesem Gegensatz zwischen der Betonung der Fähigkeit, Probleme zu lösen , und der, Erfahrungen zu verwenden , kam in diesen Diskussionen noch eine zweite Diskrepanz zum Ausdruck. Sie ergab sich aus der Frage, ob Intelligenz eine allgemeine, in allen geistigen Leistungen zur Verwendung gelangende Fähigkeit sei, oder ob sie in der Ausbildung von Spezialfähigkeiten bestehe....All diese Gesichtspunkte haben sich als wichtig erwiesen und müssen in gewissem Umfang auch alle im Auge behalten werden. Es gibt offenbar sowohl einen übergreifenden allgemeinen Faktor als auch Spezialbegabungen; es gibt intelligente Erfahrungsberücksichtigungen neben der Fähigkeit, neue Probleme zu lösen.
Diese, die Fähigkeit, originale Lösungen neu sich stellender Probleme zu finden, ist vielleicht das, was als die Höchstleistung der Intelligenz angesehen werden muß, obwohl es offenbar viele sehr intelligente Menschen gibt, die in erster Linie "Lernen", jedoch weniger zu Originalleistungen befähigt sind."(BÜHLER,C. : S.99/100).

Diese Ausführungen weisen in eine neue Richtung : Nicht nur von "Lebensbewältigung" und von "zielgerichtetem Verhalten oder Handeln" ist die Rede, sondern zum Beispiel von Lernvermögen . Diesem Aspekt werden wir in den weiteren Ausführungen besondere Aufmerksamkeit widmen. Wir werden nämlich später feststellen, daß gerade die Fähigkeiten zum Lernen (als Teil der Intelligenz) möglicherweise eine Brücke zu den Fähigkeiten von Maschinen herstellen - nicht umsonst spricht man ja nicht nur von "intelligenten Maschinen", von "denkenden Maschinen", sondern auch von "selbst lernenden Maschinen", wenn man über

"Künstliche Intelligenz" debattiert. Wir werden noch untersuchen müssen, was es mit diesen Begriffen auf sich hat.

Zuvor jedoch noch einige Versuche der Klärung des Intelligenzbegriffs. Dabei wollen wir uns nun denjenigen Definitionen zuwenden, die schon eher als die vorangegangenen erkennen lassen, ob vielleicht Maschinen auch als "intelligente Systeme" angesehen werden könnten.

"Intelligenz ist die Fähigkeit, sich unter zweckmäßiger Verfügung über Denkmittel auf neue Forderungen einzustellen.
Diese Definition läßt sich gut auf die Maschine übertragen, sogar auf Computer, wie sie überall verwendet werden." (STEINACKER,I. : 1984, S.8).

Überträgt man diese Definition tatsächlich auf Computer, dann sind die "Denkmittel" die Speicher im Computer und die darin enthaltenen Programme und nicht Gehirn und Nervenbahnen des Menschen. Die "neuen Forderungen" sind nicht diejenigen, die aus dem Versuch der Lebensbewältigung durch den Menschen resultieren, sondern die Wünsche des Computerbenutzers, der irgendwelche Informationen erhalten möchte, oder der zum Beispiel das Ergebnis einer komplizierten Rechnung durch den Computer ausgegeben sehen will.

"Bei der nächsten Definition, die einige Komponenten der Intelligenz aufzählt, schneidet der Computer nicht so gut ab....
"Intelligenz ist die aus folgenden Komponenten zusammengesetzte Fähigkeit :"

- Die erste wichtige Komponente ist "die Abbildung der Außenwelt in einem Modell."...
- Die zweite geforderte Komponente ist "die Verknüpfung von Information, die Bildung von Invarianten und deren Speicherung."...
- Als dritte Komponente der Intelligenz wird "die Konstruktion von Algorithmen des Verhaltens und das Testen dieser Algorithmen am Modell der Außenwelt" genannt."(STEINACKER,I. : 1984,

S.9/10).

Diese etwas abstrakten Formulierungen bedeuten, daß man eine zutreffende Vorstellung von den Lebensbereichen, in denen man sich bewegt ("Außenwelt") und in denen die Reaktionen zur Lebensbewältigung sich vollziehen, erst "im eigenen Kopf" entwickeln muß. Der Farbenblinde, der ein rotes Stopplicht als grüne Aufforderung zum Weiterfahren interpretiert, verfügt in seinem Kopf über eine unzutreffende Abbildung der Außenwelt und wird deshalb spezielle Schwierigkeiten bei der Lebensbewältigung haben.

Mit der zweiten Komponente in der obigen Definition wird verdeutlicht, daß Informationen allein, also die zutreffende Abbildung der Außenwelt, nicht ausreichen, sondern diese müssen miteinander in Beziehung und gegenseitig "verrechnet" werden. Unveränderliche Sachverhalte müssen als solche erkannt und gespeichert werden, d.h. man muß sie in Erinnerung behalten, um immer wieder korrekt darauf reagieren zu können. Dies gilt beispielsweise für die Verrechnung der Entfernung eines Autos und der Geschwindigkeit, mit der es herannaht, wenn man eine Straße ungefährdet überqueren möchte. Es versteht sich, daß derartige Rechenalgorithmen gespeichert werden müssen (in Erinnerung zu behalten sind), um auch zukünftig in entsprechenden Situationen gewappnet zu sein.

Auch wenn die Computer hier nicht so gut abschneiden, dürfen wir auch in diesem Zusammenhang an Maschinen denken : Verrechnen von Informationen und Speicherung der Ergebnisse gehört zu ihren Spezialitäten - wie sie gegebenenfalls dann zu korrekten Entscheidungen gelangen, soll später besprochen werden.

Bei der dritten Komponente gibt es größere Schwierigkeiten, weil der Begriff des Verhaltens nur schwer auf Maschinen übertragbar ist. Man erinnere sich aber daran, daß bei den Aktionen der sog. Industrieroboter sehr wohl von "Verhalten"

gesprochen werden kann, denn sie reagieren auf bestimmte Bedingungen, d.h. sie agieren. Bei Computern hingegen beschränkt sich "Verhalten" meist auf die Ein- und die Ausgabe von Informationen.

Wir erkennen an dieser "Komponenten-Definition" gleichwohl eine ganze Reihe von Merkmalen, die auch auf Maschinen zutreffen könnten. Dieses Ergebnis war auch bei den anderen vorgestellten Definitionen zu verzeichnen. Trotz der zu beobachtenden Unterschiede in den Definitionen der Intelligenz verzeichnen wir auch eine Reihe von Gemeinsamkeiten.

Diese Gemeinsamkeiten in den verschiedenen Definitionen erstrecken sich insbesondere auf die Möglichkeiten des Informationserwerbs zum Zwecke der Lebensbewältigung, d.h. auf die angemessenen Entscheidungen in bestimmten Situationen. Mit dieser Überlegung wird eine Verbindung herstellbar zwischen den Fähigkeiten von Maschinen, speziell von Computern einerseits und den Möglichkeiten des Einsatzes der menschlichen Intelligenz andererseits. Gerade über diese Verbindungen muß im folgenden gesprochen werden.

Bevor wir aber diese Verbindungen weiter untersuchen, ist es sinnvoll, auf einen anderen zentralen Begriff einzugehen, der konstitutives Element der in diesem Abschnitt immer wieder angesprochenen "Lebensbewältigung" ist, nämlich den Begriff des Lebens selbst.

1.3 Leben

"Leben selbst ist ein erkenntnisgewinnender Prozeß", hat der berühmte Verhaltensforscher K.LORENZ festgestellt (WEISS,P.: 1971). Mit dieser Feststellung können wir direkt an den vorangegangenen Abschnitt anknüpfen, wo ja das Stichwort "Informationsgewinnung" eine zentrale Rolle spielte. Sammeln und Gewinnung von Informationen bedeutet ja Erkenntnisgewinn oder beinhaltet wenigstens die Absicht des Gewinns von Erkenntnissen.

K.LORENZ sagt nun nicht nur, daß Leben nur auf der Grundlage von Erkenntnissen, d.h. auf der Grundlage adäquater Informationssammlung und -verarbeitung möglich sei, sondern mehr noch, daß das Leben selbst der entscheidende Prozeß der Erkenntnisgewinnung sei, daß so also der Begriff des Lebens definiert werden könnte.

Die Erkenntnisgewinnung ermöglicht erst das Zurechtfinden in der Welt und zwar dadurch, daß Ordnungen erzeugt werden : Nur in geordneten Strukturen können wir uns zurechtfinden.

"Alles Lebendige erzeugt Ordnung, wo vordem keine war", erklärt R.RIEDL in Anlehnung an E. SCHRÖDINGER (1951) (RIEDL,R. : 1979,S.24). Diese Aussage bestätigt unser obiges Argument : Die Informationsverarbeitung durch lebende Wesen erfolgt, um Ordnung zu schaffen, ohne die das lebensbewältigende Zurechtfinden nicht möglich wäre.

Aus dieser Bemerkung kann man schlußfolgern, daß lebende Wesen (oder allgemeiner gesprochen lebende Systeme) daran erkannt werden können, ob und wie sie Informationen verarbeiten. "Wenn ein System sich reproduzieren und auch angemessen mit Energie und Informationen umgehen kann, hat es einen

Anspruch darauf, als lebendig angesehen zu werden." (SIMONS,G. : 1984, S.20).

Betrachtet man diese Aussage als zutreffend, so ergibt sich die zwingende Konsequenz, daß mit dem Begriff "Leben" nicht nur solche Dinge bezeichnet werden können, die notwendigerweise irgendetwas ausschließlich mit organischen Substanzen zu tun hätten : Es sind sehr wohl Systeme vorstellbar, die in diesem Sinne "leben", ohne daß sie aus organischen Substanzen aufgebaut wären.

Wir kennen eine ganze Reihe von Systemen, die Arbeiten erledigen, wie sie bislang nur von Menschen durchgeführt wurden (man denke an bestimmte Arten der sog. Industrieroboter), ohne daß verlangt würde, diese müßten aus organischen Substanzen bestehen. Entsprechend gilt selbstverständlich, daß Computer nicht aus Kohlen-Wasserstoff-Verbindungen aufgebaut sein müssen, um die Grundrechenarten erledigen zu können.

Wenn man als das Leben kennzeichnende Kriterien die vier Stichworte

- Struktur
- Energieverarbeitung
- Informationsverarbeitung
- Reproduktion

benennt, so treffen diese Kriterien nicht nur auf den Menschen, auf Tiere und auch auf Pflanzen zu, sondern auch auf bestimmte Maschinen.

Diese Bemerkung darf nicht zu dem voreiligen Schluß führen, daß also auch Maschinen Lebewesen seien. Die Frage, ob Computer vielleicht leben oder nicht, ist auch eigentlich nicht so sonderlich wichtig, sondern von Bedeutung ist vielmehr, was sie leisten können und was nicht. Allerdings darf nicht übersehen werden, daß immer mehr Maschinenbenutzer ihren Maschinen menschliche Eigenschaften zuschreiben ("das will mein Computer nicht").

Zudem ist festzustellen, daß mehr und mehr diejenigen Definitionen des Begriffs Leben, die zum Beipiel als wichtiges Kriterium die Biochemie des Stoffwechsels verwenden, in den Hintergrund der Betrachtung treten und stattdessen solche Definitionen häufiger verwendet werden, die das Kriterium der adäquaten Informationsverarbeitung in den Mittelpunkt stellen. Nicht zuletzt deshalb haben wir ja diesen Begriff mit in den Vordergrund der bisherigen Erörterungen gestellt. Damit gewinnt man nun allerdings Definitionen, die sowohl für biologische wie auch für künstliche Systeme zutreffen können.

Auf diese Weise wird es immer schwieriger, eine exakte Trennungslinie zwischen belebter und unbelebter Materie zu ziehen, eine Schwierigkeit, die übrigens auch schon bestand, als von Robotern, Computern oder von künstlicher Intelligenz noch nicht die Rede war.

Biologen beispielsweise wissen schon seit langem ganz genau, wie schwierig diese Trennlinie zu ziehen ist : "Ein Virus kann lebendig sein oder nicht, je nach den Kriterien, die man für wichtig hält..." (SIMONS,G. : 1984, S.25).

Was für einen Virus gilt, dürfte vielleicht auch für Maschinen gelten : Verhält sich eine Maschine unter definierten Bedingungen wie ein lebendes System, dann ist sie ein lebendes System, so könnte man argumentieren. Wie sonst könnte denn überhaupt der Unterschied zwischen lebenden und unbelebten Systemen festgestellt werden, wenn nicht an ihrem jeweiligen Verhalten ? Zeigt dieses Verhalten keine Unterschiede mehr, so entfällt offenbar die Grundlage dafür, die exakte Trennungslinie zu ziehen.

Wenn ein System also adäquate Reaktionen zum Beispiel auf der Grundlage umfangreicher Informationsverarbeitungsprozesse auf sich verändernde Umweltbedingungen zeigt, so kann es - den Überlegungen des berühmten französischen

Botanikers LAMARCK entsprechend - als lebendes System klassifiziert werden.

Wenn man die Entwicklung der Gattung Mensch unter diesem evolutionstheoretischen Gesichtspunkt betrachtet, kann man allerdings auch zu einer anderen Schlußfolgerung gelangen : Anstatt den Versuch zu unternehmen, beispielsweise Computer als lebende Wesen zu klassifizieren (ein Versuch, der - wie oben schon erwähnt wurde - übrigens nicht besonders fruchtbar ist), könnte man auch argumentieren, daß die Entwicklung von Computern ein Teil unserer eigenen, der menschlichen Evolution ist :

"Nicht alle Biologen und Anthropologen (oder Computerfachleute) würden mit dieser These übereinstimmen. Natürlich kann man ihr entgegenhalten, daß die Weiterentwicklung eines Werkzeugs etwas anderes ist als die Weiterentwicklung des Tieres, das dieses Werkzeug benutzt - ebenso wie ein Hammer eben kein Mensch ist.

Aber im gleichen Atemzug sprechen Wissenschaftler von Evolutionsschritten, wenn niedere Arten der Primaten mit Werkzeugen entdeckt werden ... Schimpansen (haben) Zweige vom Baum abgebrochen, um damit Insekten aus Verstecken herauszustochern. Würde morgen ein Schimpanse auftauchen, der mit einem selbstgefertigten Hammer ankäme, dann würde wohl die gesamte wissenschaftliche Gemeinde die Entwicklung dieses Hammers als ein Zeichen der Weiterentwicklung des Schimpansen selbst verstehen." (RITCHIE,D.: 1984, S.80). "So gesehen sind heute schon Computer integrale Bestandteile von uns, daß wir ohne sie unter Umständen nicht mehr lebensfähig wären." (ebenda, S.81).

Wenn man also bei einem ersten Versuch, den Begriff des Lebens zu definieren, zu erstaunlichen Parallelen zwischen biologisch-organischen und künstlichen Systemen gelangt, so braucht man sich nicht zu wundern. Man muß sich nur an die zentrale Bedeutung des Begriffs der Verarbeitung von

Informationen erinnern, der sowohl bei Menschen und Tieren, wie auch bei bestimmten Maschinen eine wichtige Rolle spielt.

Diese Überlegungen dürfen aber nicht darüber hinweg täuschen, daß beispielsweise das menschliche Leben durch weitere Kriterien charakterisiert wird. Ganz ohne Zweifel spielt zum Beispiel das Bewußtsein , vor allem das Selbstbewußtsein ("Ich-Bewußtsein") eine ganz entscheidende Rolle :

"Ich lebe nicht nur, sondern ich weiß, daß ich lebe. Ich weiß überdies, daß ich nicht für immer leben werde, daß der Tod unausweichlich ist. Ich besitze die Eigenschaften des Selbstbewußtseins und des Todesbewußtseins." (DOBZHANSKY,T. : 1975, S.411).

Niemand kann die Behauptung widerlegen, daß künstliche Systeme, wie etwa Computer, auch über ein derartiges Bewußtsein verfügten, aber es gibt sehr wohl eine Reihe plausibler Argumente, die dieser These widersprechen :

Machen wir uns beispielsweise klar, daß das Bewußtsein des Menschen in seiner Eigenschaft als soziales Lebewesen insbesondere durch die Wechselbeziehungen zu anderen Menschen geprägt wird (beginnend mit den ersten frühkindlichen und vielleicht auch vorgeburtlichen Prozessen der Sozialisation), so sind vergleichbare Einflüsse prägender Art bei Computern nicht zu erkennen. Ob allerdings eines schönen Tages Computer ihren eigenen "Nachwuchs" produzieren und auch sozialisieren, sei noch dahingestellt.

Dies ist wohlgemerkt kein Gegenbeweis für die obige These, weil letzten Endes vielleicht auch jegliche Sozialisation als Prozeß der Informationsverarbeitung aufgefaßt werden kann, der somit auch auf künstliche Systeme zutreffen könnte. Aber es handelt sich wohl um ein hilfreiches Argument, um zu definitorischen Trennungen zu gelangen, wenn man dies wünscht.

Der Begriff des Bewußtseins kann also, wenn auch nicht ganz befriedigend, dazu taugen, den Begriff des Lebens auf solche Systeme zu beschränken, wie wir das traditionellerweise gewohnt sind : Es entstand in der Evolution ein neuer Typ "bewußten Handelns und bewußter Bestimmtheit : diese doppelte und radikale Metamorphose erklärt uns auf einleuchtende Weise das spezifisch Originale im entscheidenden Übergang vom Molekül zur Zelle - den "Sprung" zum Leben." (TEILHARD DE CHARDIN,P.: 1981, S.83).

Man darf aber bei diesem Argument nicht darüber hinweg sehen, daß es "in sich kein Widerspruch (ist), zu sagen, ein bewußtes Wesen könne unlebendig sein; umgekehrt braucht ein lebendiges System nicht bewußt zu sein".(SIMONS,G.: 1984, S.23).

Zudem gilt : "Ein Großteil unseres zweckgerichteten Verhaltens und vermutlich auch des zweckgerichteten Verhaltens von Tieren vollzieht sich ohne Einmischung des Bewußtseins. Welche biologischen Leistungen werden dann aber vom Bewußtsein unterstützt ?
Ich schlage als eine erste Antwort vor : die Lösung von Problemen nicht-routinemäßiger Art ." (POPPER,K.R./ECCLES,J.C. : 1982, S.162/163).

"...die Rolle des Bewußtseins ist vielleicht da am klarsten, wo ein Ziel oder Zweck (vielleicht nur ein unbewußtes oder instinktives Ziel oder ein instinktiver Zweck) durch alternative Mittel erreicht werden kann und wenn zwei oder mehrere Mittel nach reiflicher Überlegung ausprobiert werden. Das ist ein Fall einer neuen Entscheidung." (ebenda).

Wir erkennen an diesen Ausführungen des großen Wissenschaftstheoretikers POPPER sofort wieder die Übereinstimmung mit den bisher vorgetragenen Überlegungen : Bewußtes Handeln, wie zum Beispiel das Entscheiden bei vorgegebenen Alternativen ist Ergebnis informationsverarbeitender

Verrechnungsprozesse. Dies bedeutet, daß wir wiederum auch künstliche Systeme im Auge behalten müssen - selbst dann also, wenn wir versuchen, den Begriff des Lebens durch Einschränkung auf "bewußtes Leben" zu reduzieren.

Der Unterschied zwischen bewußt handelnden (Entscheidungen fällenden) Lebewesen und informationsverarbeitenden künstlichen Systemen, wie zum Beispiel Computern, kann folgendermaßen beschrieben werden : Der Computer agiert unbewußt; er lacht nicht über Fehler, ärgert sich nicht über Programmunterbrechungen und kann auch nicht selbst ein Gedicht bewundern, wenn ihm ein solches gelungen sein sollte (nach HARTNELL,T. : 1984, S.1).

Man kann also feststellen, daß Maschinen vielleicht intelligent sein mögen oder unter Benutzung bestimmter Kriterien sogar als lebendig angesehen werden können, aber über Bewußtsein verfügen sie (vermutlich) nicht.

Allerdings ist - dies sei noch einmal betont - die Verwendung des Kriteriums "Bewußtsein" zur Kennzeichnung des Lebens willkürlich. "...die menschliche Rasse stammt von Dingen ab, die überhaupt kein Bewußtsein hatten...es ist nicht a priori unwahrscheinlich, daß bewußte (und mehr als bewußte) Maschinen aus denen hervorgehen werden, die heute existieren..." (SIMONS,G. : 1984, S.138).

"Verschiedentlich wird behauptet, Bewußtsein könne nur existieren, wenn bestimmte Chemikalien auf bestimmte Weise zusammenwirken ... Eine solche Behauptung beinhaltet natürlich auch, daß Computerbewußtsein unwahrscheinlich ist : Schließlich bestehen Computer nicht aus Proteinen. Die Annahme lautet alo : Eine aus Proteinen konstruierte Maschine könne eventuell Bewußtsein haben, eine aus gedruckten Schaltungen und unreinem Silizium aber nicht. Einer bestimmten semantischen Entscheidung zufolge wäre dann eine organische künstliche Maschine aus Protein

tatsächlich ein vom Menschen hergestelltes Lebewesen und keine Maschine. Manche hängen an dem Gedanken, daß Maschinen aus einem anderen Stoff bestehen als sie selbst. In einem solchen semantischen Rahmen werden Maschinen niemals bewußt sein, denn dann sind sie ja keine Maschinen mehr. Offenkundig ist das aber eine Frage von Definitionen und nicht von Tatsachen." (ebenda).

Fassen wir diese Ausführungen zusammen, so erkennen wir auch hier wieder, wie schon bei der Diskussion des Begriffs der Intelligenz, daß es unterschiedliche Möglichkeiten gibt, Leben und Bewußtsein zu definieren, so daß wir letzten Endes nur schwer und nicht ohne ein Element der Willkür entscheiden können, wie und ob biologische und künstliche Systeme definitorisch getrennt werden können. Wohl aber fallen die Gemeinsamkeiten deutlich ins Auge : Wenn bewußtes Handeln lebender Organismen beispielsweise bedeutet, daß (überlebenssichernde) Entscheidungen getroffen werden, und wenn diese Entscheidungen auf der Grundlage gesammelter Informationen zustandekommen, dann sind die Parallelen unübersehbar.

In diesem (gemeinsamen) Sinne dienen Informationen der Beseitigung von bestimmten Bereichen der Unsicherheit (und genau so definiert man diesen Begriff in der Informationstheorie (siehe SHANNON,C.E./WEAVER,W. : 1949), bzw. sie ermöglichen einen Ordnungsgewinn (über den Zusammenhang zwischen Ordnung und Bewältigung des Lebens haben wir ja schon weiter oben gesprochen; zum Zusammenhang zwischen Informationen, Ordnungsgewinn und Entropie siehe z.B. : HOFSTÄTTER,P.R. : "Informationstheorie", 1957, S.169).

Auf dieser Argumentationsbasis ist es möglich, die Handlungsentscheidungen lebender Wesen als Ergebnisse informationsverarbeitender Prozesse zu interpretieren - und so hätten wir eine Brücke mehr geschlagen zwischen Lebewesen einerseits und informationsverarbeitenden Maschinen andererseits.

Es spielt dabei keine entscheidende Rolle, daß bestimmte Informationen ererbt, andere wiederum auf der Grundlage von Erfahrungen gewonnen werden, oder daß bestimmte Informationsgruppen uns unbewußt sind, andere hingegen sehr bewußt, wenn man den Begriff der adäquaten Verarbeitung von Informationen in den Mittelpunkt der Erörterungen stellt (siehe auch : POPPER,K.R./ECCLES,J.C. : 1982, S.158).

1.4 Werkzeuge

Für die weiteren Ausführungen ist es zweckmäßig, tatsächlich davon auszugehen, daß sich intelligentes Handeln und Verhalten im Sinne der Lebensbewältigung durch adäquate Verarbeitung von Informationen auszeichnen. Genau so läßt sich ja in der Tat belegen, daß die evolutionären Prozesse zur jetzt vorfindbaren Situation geführt haben. So ist erklärbar, daß diejenigen Lebewesen, die wir kennen, heute die Erde bevölkern (und keine anderen) - die Gattung Mensch natürlich eingeschlossen.

Jene Gattungen haben die Oberhand gewonnen, die sich optimal an die herrschenden Bedingungen anpassen konnten. Dem Menschen ist dies ganz besonders gut deshalb gelungen, weil die informationsverarbeitenden Prozesse schon sehr früh dazu geführt hatten, daß er sich geeignete Werkzeuge zur Erledigung bestimmter Aufgaben schuf und sich ihrer geschickt bediente.

Daß dies allerdings kein auf den Menschen beschränkter evolutionärer Prozeß war, belegt überzeugend das Experiment zur Überprüfung der Denkfähigkeit undressierter Schimpansen : Man "erdachte eine geniale Methode, die Tiere in Situatioen zu bringen, bei denen praktische Probleme durch Denken zu lösen waren; dadurch, daß mit Schimpansen gearbeitet wurde, war der Sprachfaktor im Denkprozeß ausgeschaltet - Schimpansen haben keine Sprache im menschlichen Sinn. Ziel der Versuche war, ein eventuell vorhandenes technisches Denken zu ermitteln. Ein solches Denken konnte auch tatsächlich festgestellt werden. Die begabtesten unter den Schimpansen, als erster Sultan, der heute geradezu

als "historische Persönlichkeit" gelten kann, waren fähig, sich Werkzeuge herzustellen, mit denen sie zu Gegenständen gelangen konnten, die ihnen direkt nicht erreichbar waren. In dem ersten klassischen Experiment wurde außerhalb des Käfigs von Sultan eine Banane so hingelegt, daß er sie mit der Hand nicht greifen konnte. Im Käfig lag ein Stock. Und nun hatte Sultan den Einfall, diesen Stock zum Heranholen der Banane zu benutzen.
Was war geschehen ? Sultan hatte die begehrte Banane durch die Stäbe des Käfigs gesehen und die Entfernung, in der sie lag, als zu weit eingechätzt. Dann sah er den Stock. In einer unzweifelhaft schöpferischen geistigen Leistung sah er in seiner Vorstellung den Stock als Verlängerung seines Armes - als ein bis zu der Banane hin reichendes Werkzeug." (BÜHLER,C. : 1962, S.102).

In ähnlicher Weise wird wohl die Entwicklung erster Werkzeuge beim Menschen auch verlaufen sein.

Wir halten es für sinnvoll, dieser Frage der Werkzeuge einen eigenen Abschnitt zu widmen, weil wir Computer - und sie sind je letzten Endes das zentrale Thema dieses Buchs - auch als Werkzeuge des Menschen ansehen können; dieser Gedanke klang ja auch schon im letzten Abschnitt an.

Allerdings geht es nun nicht um solche Werkzeuge, die von der Last körperlicber Tätigkeiten und Anstrengungen befreien oder wenigstens zur Unterstützung bei derartigen Arbeiten dienen, sondern um Werkzeuge, die uns von der Last geistiger Tätigkeiten mehr oder weniger befreien sollen.

So wie der Faustkeil, der Hammer, der Hebel, das Rad, der Flaschenzug, die Dampfmaschine, das Auto, die Elektrotechnik und die Nukleartechnologie uns in der Evolution vorangebracht haben (ob immer nur zum Nutzen der Menschheit und der übrigen Welt sei dahingestellt), so gilt dies entsprechend auch für

Werkzeuge, die zur Unterstützung oder zur Übernahme geistiger Aktivitäten geeignet sind.

Wir wollen mit dieser Argumentation nicht die oft zu hörende Bemerkung stützen, Computer seien nur Werkzeuge, sondern zeigen, daß diese Ausdrucksweise der hier zur Debatte stehenden Problematik nicht gerecht wird. "Die Funktion des Wörtchens "nur" in der obigen Aussage, soll zu dem Schluß verleiten, daß der Computer in keinem fundamentalen Sinne sehr wichtig sein kann, da Werkzeuge an sich nicht sehr wichtig sind. Ich habe behauptet, daß Werkzeuge die Rekonstruktion der Wirklichkeit in der Phantasie des Menschen formen und deshalb an der Ausbildung der menschlichen Identität beteiligt sind." (WEIZENBAUM,J. : 1978, S.213).

Aus dieser Bemerkung WEIZENBAUMs wird deutlich, daß Werkzeuge ganz allgemein mehr sind, als nur Gegenstände, derer man sich bedient. Sie sind vielmehr ein notwendiges Bindeglied zwischen der uns umgebenden Welt und dem Bild der Welt , wie es in unserem Kopf existiert. Werkzeuge sind sowohl Teil der Welt wie auch Produkt unserer geistigen Tätigkeit, im Versuch, die Welt zu begreifen und sie zu gestalten.

Wie entstehen derartige Werkzeuge ?

Üblicherweise unterscheidet man in diesem Zusammenhang Performanz- und Simulationsmodus . Den Unterschied zwischen beiden Modi erläutert man oft am Beispiel des Fliegens :

Alle frühen Versuche, das Prinzip des Fliegens zu erkennen, bzw. dem Menschen das Fliegen beizubringen, beruhten auf der Nachahmung des Vogelflugs. Diese Versuche scheiterten bekanntlich - von IKARUS bis LILIENTHAL. Es handelte sich dabei ganz offenbar um Versuche, das Problem des Fliegens im Simulationsmodus zu lösen.

In der zweiten Hälfte des vergangenen Jahrhunderts ging man dann zum Performanzmodus über : Die

Aufgabenstellung lautete jetzt, Flugmaschinen zu bauen, unabhängig davon, auf welchen Prinzipien sie beruhten, die man bei diesen Konstruktionsversuchen entdecken würde.

"In diesem Zusammenhang sollte vielleicht noch ein dritter Operationsmodus erwähnt werden : der Theoriemodus . Noch vor dem Bau funktionierender Flugzeuge gab es große Aerodynamiker ... Während sich jedoch der Aerodynamiker für die Theorie an sich interessiert und funktionierende Flugzeuge als bloße Modelle seiner Theorie betrachtet, sieht der Flugzeugkonstrukteur die Theorie lediglich als eine von vielen Quellen, deren Gedanken ihm behilflich sein können, seine Maschinen noch leistungsfähiger zu machen." (WEIZENBAUM,J. : 1978, S.220).

Auch wenn wir einen Computer als Werkzeug auffassen, das benutzt wird, um beispielsweise arithmetische Operationen durchzuführen, sollten wir diese Aspekte - insbesondere im Hinblick auf das noch zu diskutierende Stichwort von der Künstlichen Intelligenz - im Auge behalten :

Soll mit dem Computer konkret die Tätigkeit des menschlichen Gehirns nachgeahmt werden (also Simulationsmodus), oder geht es eher darum, funktionsfähige Geräte zu bauen, solche also, die etwa 2 und 2 addieren können, auch wenn sie vielleicht nach ganz anderen Prinzipien arbeiten wie das Gehirn (Performanzmodus) ? Ist ein Computer ein Modell, das einer Theorie des Gehirns entspricht, oder ist diese Theorie nur Hilfsbasis zur Entwicklung des Computers ?

Diese Fragen zeigen, daß auch der Computer als "Mittler" zwischen menschlichem Geist und realer Welt fungiert, eine wechselseitige Beziehung herstellt, gleichzeitig also Teil der realen Welt, wie auch Teil unseres Gehirns ist - in diesem Sinne vielleicht einer gut funktionierenden Handprothese vergleichbar.

Das Stichwort "Werkzeug" bezeichnet also nur

unzureichend den Sachverhalt, um den es hier geht; wenn schon, dann müßte von einem "Werkzeug zur Gehirnunterstützung" gesprochen werden. Es kommt dabei entscheidend nicht mehr darauf an, zu verstehen, wie ein derartiges Werkzeug funktioniert, sondern darauf, es adäquat einzusetzen.

Allerdings machen wir uns dann von diesem Werkzeug in gewisser Weise abhängig : Wir werden zum Diener eines Gesetzes, "das wir nicht kennen können, also die Diener eines unberechenbaren Gesetzes. Und das ist es dann, was uns so unruhig macht." (WEIZENBAUM,J. : 1978, S.67).

Auch wenn man aber die Funktionsweise eines Werkzeugs im einzelnen nicht versteht, so bleibt doch richtig, was oben skizziert wurde, nämlich die Wechselseitigkeit des Einflusses zwischen Mensch, Werkzeug und Realität.

Wenn man sorgfältiger der Frage nachgeht, welches diese wechselseitigen Beziehungen sind, so muß man sich mit einer vergleichenden Betrachtung von Gehirn und Computer befassen. Dabei interessieren weniger die physiologischen beziehungsweise die technologischen Bedingungen (das würde viel zu weit führen) als die Gemeinsamkeiten und die Unterschiede in den Ergebnissen der jeweiligen Aktivitäten.

"... eine Mikroanalyse der Gehirnfunktionen (ist) für ein Verständnis der Denkprozesse so wenig sinnvoll, wie eine entsprechende Analyse der einen Computer durchfließenden Impulse für das Verständnis des Programms, das der Computer gerade abarbeitet. Untersuchungen dieser Art würden einfach auf dem falschen theoretischen Niveau ansetzen." (WEIZENBAUM,J. : 1978, S.184).

Der Versuch eines Vergleichs muß an einer anderen Ausgangsbasis ansetzen, nämlich an den Überlegungen der vorangegangenen Abschnitte, wo mehrfach festgestellt wurde, daß das menschliche Gehirn sehr wohl als informationsverarbeitende

Maschine bzw. Leben als informationsverarbeitender Prozeß verstanden werden können. Selbst diejenigen Autoren, die prinzipiell die Vergleichbarkeit von menschlichem Gehirn und Computern in Frage stellen, können diese These noch akzeptieren :

"Ich werde im folgenden die Position vertreten, daß es kein Fehler ist, den Menschen als Verarbeiter von Informationen (oder auch als irgend etwas anderes) anzusehen oder ihn unter dieser Perspektive zu verstehen zu versuchen, allerdings unter der Voraussetzung, daß wir niemals so tun, als könne irgendeine einzelne Perspektive den ganzen Menschen erfassen. Daß wir den Menschen als informationsverarbeitendes System sehen, bedeutet für sich genommen noch keine Dehumanisierung, sondern kann im Gegenteil insofern zu seiner Humanität beitragen, als es ihm zu einem vertieften Verständnis eines spezifischen Aspekts seiner menschlichen Natur verhilft." (WEIZENBAUM,J. : 1978, S.190).

Unter diesem Aspekt ist es also wichtig, festzuhalten, daß Prozesse, die im menschlichen Gehirn ablaufen, wenigstens teilweise durch Maschinen, also zum Beispiel durch Computer simuliert werden können. "Der Öffentlichkeit ist weitgehend unklar - obwohl sie vom Gegenteil fest überzeugt ist - , was es heißt, daß prinzipiell jedes effektive Verfahren von einem Computer durchgeführt werden kann. Da der Mensch, die Natur und selbst die Gesellschaft mit Verfahren arbeiten, die zweifellos in der einen oder der anderen Hinsicht "effektiv" sind, folgt daraus, daß ein Computer zumindest den Menschen, die Natur und die Gesellschaft in allen verfahrensmäßigen Aspekten imitieren kann." (ebenda, S.210).

Computer können also in der Tat als Werkzeuge angesehen werden, die in der Lage sind, gedankliche Problemlösungswege zu simulieren und das heißt letztlich nichts anderes, als daß sie Probleme lösen können. Dies bedeutet aber nicht gleichzeitig, daß <u>alle</u> gedanklichen Prozesse durch

Computer simuliert werden könnten.

Der einzelne Mensch ist ja vor allem auch durch die Probleme definiert, denen er gegenübersteht, und die aus seinen biologischen und emotionalen Bedürfnissen entstehen. Insbesondere sind auch die Sozialisationsbedingungen, die Konfrontation mit anderen Menschen für das "Menschsein" entscheidend wichtig. "Kein anderer Organismus, und erst recht kein Computer, kann dazu gebracht werden, ausschließlich menschliche Probleme auf menschliche Weise zu lösen. Und da mit Ausnahme einer kleinen Anzahl formaler Probleme der Bereich der menschlichen Intelligenz durch das Menschsein des Menschen bestimmt ist, muß jede andere Intelligenz - wie hoch auch immer - dem menschlichen Bereich fremd sein." (ebenda, S.295).

Diese Aussagen sind sehr deutlich. Sie besagen schlicht, daß nicht alles, was sich im menschlichen Gehirn vollzieht, durch Computer simuliert werden kann - und manche Autoren halten es für völlig ausgeschlossen, daß sich dies je ändern könnte. Gleichwohl ist es sicherlich nicht falsch, den Computer - genauso wie zum Beispiel das Rad - als nützliches Werkzeug zu betrachten und ihn entsprechend zu verwenden.

Dieser "werkzeughafte" Einsatz des Computers geschieht ja nun auch in zunehmendem Maße, so daß er schon heute unentbehrlich geworden ist : "Der Computer wird zum unentbehrlichen Bestandteil jeder Struktur, sobald er so total in die Struktur integriert ist, so eingesponnen in die verschiedensten lebenswichtigen Substrukturen, daß er nicht mehr herausgenommen werden kann, ohne unweigerlich die Gesamtstruktur zu schädigen." (WEIZENBAUM,J. : 1978, S.49/50).

WEIZENBAUM beschwört hier eine große Gefahr : Wenn sich herausstellen sollte, daß der Computereinsatz per Saldo mehr schadet als nützt, dann ist es kaum mehr möglich, die ökonomischen, gesellschaftlichen und politischen Strukturen wieder so zu verändern, daß man ohne Computer auskäme. Das "Werkzeug"

Computer hat sich unentbehrlich gemacht, es hat sich verselbständigt.

Es soll hier nicht diskutiert werden, ob diese Verselbständigung von Vorteil oder von Nachteil für uns ist - daß sie nicht ganz unproblematisch ist, liegt auf der Hand.

Es hat sich eine enge Verzahnung ergeben zwischen dem Bedürfnis des menschlichen Gehirns, die Welt zu erklären und Geräte zur leichteren Lebensbewältigung zu schaffen einerseits und den unter diesen Zielvorgaben entstandenen Geräten selbst andererseits. Unter vergleichenden Aspekten ergibt sich deshalb ein direkter Zusammenhang zwischen dem menschlichen Gehirn und dem Computer folgender Art :

"In der Annahme, daß das Ziel, intelligente Computer zu schaffen, nicht unabhängig ist vom Bestreben, menschliche kognitive Leistungen zu erklären, wird versucht, Korrespondenzen herzustellen zwischen (abstrakten) Software - Strukturen im Computer (bzw. deren funktionaler Interpretation) und postulierten (abstrakten) Mentalstrukturen des Menschen. Beide, Mensch und Computer, werden hier als Realisation derselben theoretischen Vorstellungen angesehen." (KOBSA,A. : 1984, S.107/108).

Unter diesem Gesichtspunkt verliert der Computer etwas von seinem instrumentalen Charakter als Werkzeug : Es geht nicht mehr darum, immer "schlauere" Computer zu konstruieren, sondern um das "Verstehen intelligenter Prozesse unabhängig von ihrer speziellen physischen Realisierung." (GOLDSTEIN,I./PAPERT,S. : 1977).

Wir erinnern in diesem Zusammenhang an die obige Erörterung um Performanz-, Simulations- und Theoriemodus : Nicht mehr die Simulation menschlicher Gehirnprozesse interessiert heute primär, als vielmehr der Versuch, zu einem generellen Verständnis problemlösender Prozesse zu gelangen : "Solange man Flugzeuge gebaut hat, die

in Nachahmung der Vögel flatternde Flügel besaßen, waren die Resultate recht kläglich. Die heutigen Flugzeuge flattern nicht und übertreffen dennoch sämtliche Vögel in der Geschwindigkeit, weil sie materialadäquat konstruiert sind. Computergerechte Programme ... müssen keineswegs ähnlich unseren Denkvorgängen ablaufen." (TRAPPL,R. : 1984, S.205).

Computer sind nicht mehr einfach nur Werkzeuge, sondern sie sind ein Versuch, Denken zu verstehen. Allein schon deshalb ist es erforderlich, den folgenden Abschnitt dem Begriff "Denken" zu widmen. Diese Überlegungen bestätigen erneut, daß wir vor einem großen Problem stehen :

"Wenn Maschinen richtig funktionieren, so folgen sie nicht einfach einem Gesetz; sie sind Verkörperungen von Gesetzen ... (von Gesetzen, die) wir kennen und anwenden wollen." (WEIZENBAUM,J. : 1978, S.66). Und daraus ergibt sich konsequenterweise, wenn wir die Funktionsweise eines modernen Rechners nicht hinreichend verstehen : "Wenn wir von dieser Maschine abhängig sind, so sind wir Diener eines Gesetzes geworden, das wir nicht kennen können, also die Diener eines unberechenbaren Gesetzes. Und das ist es dann, was uns so unruhig macht" (ebenda, S.67).

Diese Aussage gilt ganz generell, also auch für solche Maschinen, die mit der Übertragung von Informationen und nicht mit der Ersetzung körperlicher Kräfte zu tun haben. Dies gilt demnach auch für jene Maschinen, die uns von bestimmten Denkprozessen entlasten sollen, also für Computer.

Folgt aus all dem, was gesagt wurde auch, daß Maschinen, die Denkprozesse ersetzen, denken können ?

1.5 Denken

Kein Computer, so stellt man häufig nicht ohne Befriedigung fest, wird jemals in der Lage sein, den Hyperion von HÖLDERLIN zu schreiben oder das Violinkonzert von BEETHOVEN zu entwerfen, denn - so sagt man - ein Computer kann ja nicht denken.

Ich muß bekennen, daß auch mir kein klassischer Roman gelingen will und auch kein Beethoven - Konzert ist unter meiner Feder entstanden. Folgt daraus, daß auch ich nicht denken kann ? Offenbar (oder hoffentlich) nicht.

Der Begriff des Denkens muß also anhand anderer Kriterien definiert werden. HOFSTÄTTER hat sehr anschaulich beschrieben, was denn überhaupt Denken sei : "In Situationen, für deren Bewältigung wir weder ererbte Instinkthandlungen ... noch auch mehr oder minder automatische, zur Gewohnheit gewordene, erlernte Verhaltensweisen ... bereithalten, pflegen wir unser Tun für eine Weile zu unterbrechen, um uns das weitere Vorgehen zu überlegen. Was in dieser Pause geschieht, bezeichnet man als Denken. Aus der Innensicht des Menschen ergibt sich, daß dabei verschiedene Möglichkeiten und Wege des Agierens bezüglich ihrer Vor- und Nachteile gegeneinander abgewogen werden." (HOFSTÄTTER,P.R. : 1957, S.86).

Dieser Denkprozeß ist übrigens nicht nur eine menschliche Vorgehensweise, sondern findet sich auch bei Tieren : "Manche Verhaltensbiologen nehmen jedenfalls keinen Anstoß daran, auch bei Tieren bisweilen von einem "Nachdenken" zu sprechen, so z.B. ... K.LORENZ." (ebenda, S.87).

Die Unterschiede zwischen menschlichem und tierischen Denken werden durch den Begriff des

Bewußtseins verdeutlicht, der in anderem Zusammenhang schon erwähnt wurde : "Ich habe kaum Zweifel daran, daß Tiere bewußt leben und vor allem, daß sie Schmerzen empfinden, und daß ein Hund hocherfreut sein kann, wenn sein Herr zurückkehrt. Aber ich vermute, daß nur ein menschliches, der Sprache mächtiges Wesen über sich selbst reflektieren kann. Ich glaube, daß jeder Organismus ein Programm hat. Ich glaube aber auch, daß nur ein menschliches Wesen sich einiger Teile dieses Programms bewußt sein und sie kritisch revidieren kann." (POPPER,K.R. / ECCLES,J.C. : 1982, S.183).

Entsprechend äußert sich Pierre TEILHARD DE CHARDIN : "Ganz gewiß, das Tier weiß. Aber sicher weiß es nicht, daß es weiß . Sonst hätte es Erfindungen seit langem gehäuft und einen systematischen geistigen Aufbau vollbracht, der unserer Beobachtung nicht entgehen könnte." (TEILHARD DE CHARDIN,P. : 1981, S.166).

Die Frage, die uns hier nun beschäftigt, lautet natürlich : Können auch Computer denken ?

Es wird den Leser nicht verwundern, daß diese Frage sehr unterschiedlich beantwortet wird, und daß sich in der Regel zunächst Widerspruch erhebt, der vielleicht folgendermaßen umrissen werden kann :

Selbst wenn in einem Computer Prozesse ablaufen, deren Ergebnisse an die Ergebnisse menschlichen Denkens erinnern oder sogar direkt mit ihnen vergleichbar sind, so gilt doch, daß Computer nur "denken" (wenn überhaupt) auf der Grundlage vom Menschen vorgegebener Programme.

Gegen dieses Argument können aber auch Einwände erhoben werden : Auch das menschliche Denken vollzieht sich auf der Grundlage von "Programmen", die "von außen" vorgegeben werden - entweder ererbt oder von anderen Menschen oder der umgebenden Umwelt "einprogrammiert"; von Eltern, Lehrern, Nachbarn, Freunden, von allen

Umweltbedingungen, denen der Mensch ausgesetzt ist.

Das folgende Argument kann ebenfalls verwendet werden, um die These, daß der Computer letztlich nur vom Menschen abhängt, anzugreifen : "Der naive Leser wird nun denken, daß doch gewiß derjenige, der das Programm eingeschrieben hat, das System verstehe ? - Dem ist nicht so, weil es diesen einen nicht gibt : vielleicht sind es zehn, zwanzig oder hundert Programmierer, die an diesem System schon gearbeitet haben, und zwar zu verchiedenen Zeiten und an verschiedenen Orten, so daß keiner weiß, was der andere gemacht hat. So etwas wie eine strenge Architektur, an die man sich halten könnte, ist gar nicht vorhanden." (WEIZENBAUM,J. : 1984, S.94/95).

Wo ist also der Unterschied ?

Wenden wir uns einmal zunächst der Frage zu, was "Denken" beim Computer überhaupt bedeuten könnte : Bedeutet das Abarbeiten vorgegebener Programme, also das "Nachahmen" vorgegebener Denkprozesse "Denken" ? Vermutlich nicht. Allerdings ist nur schwer einzusehen, wieso ein Schachprogramm, das seinen Schöpfer dauernd schlägt (und solche gibt es inzwischen), nur nachahmt (SIMONS,G. : 1984, S.10). - ganz zu schweigen davon, daß mein Hund, der - so K.LORENZ - sehr wohl Denkprozesse zustandebringt, weit schlechter Schach spielt.

Das Schachbeispiel kann uns zu dem folgenden Ergebnis führen :

Wenn ein Computer sich so verhält, als könne er denken, dann denkt er ! Präziser : Wenn nicht unterscheidbar ist, ob ein Computer <u>Denken simuliert</u> oder ob er <u>denkt</u> , dann spricht nichts gegen die Annahme, daß er denkt.

Man kann dieses Problem auch von einer anderen Seite her anpacken :

Wenn wir das Gehirn als informationsverarbeitende

"Maschine" ansehen (der schon mehrfach zitierte J.WEIZENBAUM betont allerdings mit Nachdruck, daß das Gehirn mehr als das ist), dann ergibt sich daraus die Möglichkeit, daß wir eine Maschine entwerfen können, die wie ein Mensch denkt (KENT,E.W. : 1978). Die Analogien zwischen dem menschlichen Gehirn und dem von eben diesem Gehirn entworfenen und allein schon deshalb ihm ähnlichen Computer liegen in der Tat auf der Hand :

Der Computer erledigt informationsverarbeitende Prozesse auf binärer Basis, d.h. letzten Endes durch Schließen und Offenlassen von Stromkreisen. Und genau auf dieser binären Grundlage funktionieren auch Impulsübertragungen in den Gehirnzellen (siehe dazu die detaillierte Darstellung bei J.C.ECCLES, 1978).

"Es gibt interessante Ähnlichkeiten zwischen Kohlenstoff (der in den herkömmlichen Lebensformen für jede Art von lebendem Gewebe, auch für das Gehirn gebraucht wird) und Silizium (das für alle Arten von Computerschaltkreisen verwendet wird, auch für die logischen und arithmetischen). Kohlenstoff und Silizium sind zwei der sehr wenigen Elemente, aus denen hochkomplizierte Moleküle (Makromoleküle) erbaut werden können : Man weiß, daß solche komplexen Moleküle für die Stoffwechselvorgänge notwendig sind, die in den anerkannten Lebensformen stattfinden." (SIMONS,G. : 1984, S.131).

Die interne Informationsübertragung im Computer ist vielleicht also durchaus vergleichbar mit derjenigen im menschlichen oder tierischen Gehirn - und weitere Analogien treten hinzu : Computer können Informationen speichern, handhaben und abrufen, sie können Probleme lösen und Entscheidungen treffen - alles dies sind typische Denkakte.

"Bereits in den 50-er Jahren erschienen ... etwa 1000 Abhandlungen, die sich mit dem Thema "Computer und Denken" beschäftigten. Hauptdiskussionspunkt dabei war die Frage,

inwieweit es statthaft sei, dem Computer umgangssprachliche mentale Begriffe, wie etwa Denken, Verstehen, Bewußtsein, Gefühl u.ä. zuzusprechen." (KOBSA,A. : 1984, S.101).

Lassen wir einmal einen Autor zu Wort kommen, der anschaulich zu machen versucht, daß vom "denkenden Computer" nicht die Rede sein kann :

"Zu der Behauptung, eines Tages könnten schlaue Computer ebenso denken wie Sie und ich, hegt (SEARLE) keine positive Einstellung. Wenn ein Computer "denkt", so SEARLE, dann denkt er noch lange nicht, sondern er führt nur etwas aus, von dem die Menschen denken, es sei Denken. Zur Veranschaulichung läßt SEARLE das Bild eines riesigen Computers vor unseren Augen entstehen, der wie ein menschliches Hirn arbeiten soll, aber aus Milliarden leerer Bierdosen zusammengesetzt ist. Jede Bierdose verkörpert ein Neuron. Alle Dosen sind untereinander durch Hebel verbunden und werden durch eine Windmühle angetrieben. Nehmen wir nun einen Gedanken, der uns häufiger kommt : "Ich habe Durst." Will SEARLEs Bierdosencomputer diesen Satz nach-denken, dann müssen wir uns vorstellen, wie all die Dosen und Hebel losrattern und klappern, um die Bewegungen der neuronalen Interaktionen im Gehirn zu imitieren. Ist die Operation abgeschlossen, dann kommen die Dosen zur Ruhe, das Geklappere hört auf und eine Dose plumpst aus einer Öffnung heraus. Auf ihr steht : ICH HABE DURST. In einem 1982 erschienenen Artikel fragt SEARLE : "Glaubt irgend jemand, daß dieser Apparat tatsächlich in dem Sinne durstig ist, wie Sie oder ich es sind ?"" (RITCHIE,D. : 1984, S.193).

Diese Ausführungen klingen zugegebenermaßen ein wenig trivial, und es ist deshalb sinnvoll, die Unterschiede zwischen Gehirn und Computer - was das Denken anbelangt - etwas präziser darzustellen. J.WEIZENBAUM gelingt dies mit einer sehr einprägsamen Feststellung :

"Das menschliche Vorstellungsvermögen muß in der

Lage sein, die Grenzen physikalischer Gesetze zu überschreiten, ehe es diese Gesetze überhaupt verstehen kann." (WEIZENBAUM,J. : 1978, S.155). Es dürfte einleuchten, daß ein Computer hingegen die Grenzen, die durch physikalische Gesetze vorgegeben sind, in keinem Fall überschreiten kann.

Daraus ergibt sich, daß es Denkakte gibt, die ausschließlich dem Menschen vorbehalten sind, oder - vorsichtiger ausgedrückt - ihm zumindest vorbehalten bleiben sollten.

Dies ändert aber nichts an der Aussage, daß gewisse Parallelen unübersehbar sind. Sie zeigen sich allein schon darin, daß die menschlichen Denkakte mit davon abhängen, welche Vorräte an Informationen im Gedächtnis des Menschen gespeichert sind. Dies gilt in entsprechender Weise auch für Computer, deren Leistungsfähigkeit ebenfalls insbesondere von der Quantität und der Qualität der gespeicherten Informationen abhängt.

Ohne Gedächtnis wäre jede Informationsverarbeitung zum Scheitern verurteilt oder zumindest recht ineffizient, weil ja bestimmte Denkprozesse immer wieder neu durchdacht werden müßten, könnte man ihre Ergebnisse nicht speichernd bewahren. Nur wenn auf erworbenem Wissen schon aufgebaut werden kann, kommt der so wichtige Prozeß des Lernens in Gang.

"Lernen" im Sinne der Verwertung bereits gespeicherter Informationen und Erweiterung des Informationsvorrats können auch Computer - also auch hier wieder eine bemerkenswerte Parallele : "Computer lernen das Lernen. Das leisten sie auf unterschiedliche Weise. Heuristische und adaptive Programme werden als eine Möglichkeit angesehen, durch die Computer aus Erfahrung lernen können : Dies wird etwa gestützt von den Ergebnissen, die mit heuristischen Schachprogrammen erzielt wurden - sie können aus Erfahrung so viel lernen, daß sie ihre menschlichen Programmierer bei weitem übertreffen. ... Computer lernen, (die)

verschiedenen Arten des Lernens zu erforschen, und entwickeln gleichzeitig Möglichkeiten, Informationen zu speichern und diese im Licht neuer Erfahrung zu modifizieren. ... Computer reagieren wie Menschen auf die Schwierigkeiten des Lebens unter anderem damit, daß sie lernen und so ihre Lage bewältigen." (SIMONS,G. : 1984, S.157).

Wir können sogar davon ausgehen, daß Computer "effektiver" lernen als beispielsweise der Mensch : Letzterer vergißt häufig wieder, was er einmal gelernt hat, ein Computer nicht - sofern die Speichermedien in Ordnung sind. Erinnerungen, die das menschliche Gehirn aufgrund früheren Informationserwerbs bewahrt, verändern sich, verfälschen sich manchmal sogar. Nicht so beim Computer. Ihm kann es nicht passieren, daß bestimmte Informationen im Zeitablauf restlos verloren gehen, oder daß sie sich mehr oder weniger stark verändern.

Der Leser bemerkt, daß wir hier nur über solche Gedächtnisinhalte sprechen, die willentlich abrufbar sind und nicht über solche, die anderer Natur sind (nicht willentlich abrufbar, wie zum Beispiel Reflexe, manuelle Fähigkeiten u.ä.). Dies ändert jedoch nichts an der Gültigkeit des obigen Arguments.

Die Fachleute auf diesem Gebiet unterscheiden häufig zwei Arten von Gedächtnis :

"(1) Hirnspeicherungsgedächtnis , das in den Datenbanken des Gehirns festgehalten wird, speziell in der Großhirnrinde;

(2) Erkennungsgedächtnis , das vom selbstbewußten Geist bei seiner Prüfung der Abrufe aus den Hirnspeicherungserinnerungen angewandt wird." (POPPER,K.R./ECCLES,J.C. : 1982, S.479).

Wenn wir Vergleiche zwischen dem menschlichen Gehirn und dem Gedächtnis eines Computers (und den darauf aufbauenden Denkprozessen) anstellen, ist es zweckmäßig, diese Unterscheidung zu verwenden :

Ein "denkender" Computer benötigt nicht nur eine Datenbank, sondern er benötigt ebenfalls das sog. Erkennungsgedächtnis oder ein Pendant dazu.

Bei der Erörterung der einzelnen Computerprogramme zur Künstlichen Intelligenz werden wir diesem Sachverhalt wieder begegnen.

Dieses Problem des Erkennens ist deshalb so wichtig, weil ganz generell gesagt werden kann, daß nur derjenige zielgerichtet denken kann, der verstanden hat, und daß nur derjenige verstanden hat (ob Mensch, Tier oder Maschine), der erkennt.

Das menschliche Denken nun vollzieht sich in Bahnen, die durch die Möglichkeiten der Sprache festgelegt sind. Rationales Denken ohne Sprache ist kaum vorstellbar. "Unsere Gedanken sind so eng mit Worten verknüpft, daß wir nur schwer ohne sie denken können. Das, was wir "Gedanke" nennen, wäre ohne eine Art von Sprache in der Tat unmöglich, denn jene bietet uns die Symbole für alles, was wir sehen und hören und denken und fühlen." (RITCHIE,D. : 1984, S.167).

Erkennen als Voraussetzung des Denkens bedeutet deshalb in erster Linie Spracherkennung - und gerade hier stehen natürlich Computer vor einer Reihe schwierigster Probleme, weil die menschliche Sprache häufig nicht eindeutige Konstruktionen aufweist :

Der Satz

"Ich sah einen Mann auf dem Hügel mit dem Teleskop"

läßt drei verschiedene Deutungen zu und kein Computer wird in die Lage versetzt werden können, zu erkennen, welche der drei Bedeutungen im konkreten Fall gemeint ist. Erschwert wird diese Problematik durch die recht oft zu verzeichnende Mehrdeutigkeit einzelner Begriffe.

"Es ist deshalb kaum zu glauben," schreibt

J.WEIZENBAUM, "daß viele meiner Kollegen es nicht begreifen können, daß dem Computer hier Grenzen gesetzt sind : die natürliche Sprache wird er nie verstehen, und zwar weil es ein unbedingtes Verstehen nicht gibt. Verstehen setzt immer ein Wesen voraus, das etwas in seinem Zusammenhang erkennen kann, und wieweit es das kann, hängt von seiner eigenen Geschichte ab : jeder Mensch wird deshalb das gleiche immer ein wenig anders verstehen, und so ist es unmöglich, den absoluten Sinngehalt überhaupt zu erfassen." (WEIZENBAUM,J, : 1984, S.101).

"Eine Geschichte aber, wie jeder einzelne Mensch sie hat, kann der Computer nicht haben und deshalb auch kein Verständnis im eigentlichen Sinne des Wortes.
"Was macht es schon aus", sagen meine Gegner, "ob der Computer eigene Erlebnisse hat oder ob ihm diese Erlebnisse mittels eines Magnetbands einverleibt werden !"
Aber das würde nun bedeuten, daß alle menschlichen Erfahrungen in Sprache umgesetzt werden können. Dem ist nicht so. Ich glaube sogar, daß wir das wenigste von dem, was wir als Menschen wissen und erfahren, auch aussprechen können." (ebenda).

Diese Aussagen schließen natürlich nicht aus, daß Erkennen auf anderen Ebenen - außerhalb der Spracherkennung also - schon eher durch Computer geleistet werden könnte : So hat man "vorausgesagt, daß in den achtziger Jahren Roboter mit verschiedenartigen Sinnen ausgestattet sein werden, um Zustand und Einzelheiten ihrer Umgebung zu registrieren. Kontrollsysteme werden entsprechend ausgebaut, um sensorische Informationen so zu nutzen, daß Roboter zielgerichtet handeln können..." (SIMONS,G. : 1984, S.108).

Dies ändert aber alles nichts an dem Tatbestand, daß der korrekte, d.h. der verstehende Umgang mit der Sprache eine notwendige (wenn auch nicht hinreichende) Bedingung für Computeraktivitäten ist, die mit dem Begriff des menschlichen Denkens

verglichen werden sollen.

Die Sprache ist ein wesentliches Charakteristikum des Menschen (nichtsprachliche Kommunikation, wie sie bei Tieren möglich ist, soll hier nicht als "sprachliche Kommunikation" bezeichnet werden). Sprache in diesem Sinn ist nur innerhalb eines Kontextes verständlich, der dem Menschen, nicht aber dem Computer mitgegeben ist. Die Frage, ob und wie gegebenenfalls einem Computer dieser Kontext mitgeteilt werden könnte, ist bislang weitgehend ungelöst und eine große Zahl wissenschaftlicher Veröffentlichungen befaßt sich gerade mit diesem speziellen Problem.

Es darf allerdings nicht übersehen werden, daß in bestimmten Bereichen "die Unterscheidungen zwischen Sprachen und deren Verkörperungen durch Maschinen verschwinden." (WEIZENBAUM,J. : 1978, S.144). In diesen Bereichen kann der Computer all das, was auch der Mensch kann, mit dem Unterschied, daß der Computer die jeweilige Aufgabe zuverlässiger erledigt. Alle Programmierer, die lange genug mit Computern gearbeitet haben, werden dies bestätigen.

"Erkennen" durch den Computer ist also, so können wir zusammenfassend feststellen, nicht ausgeschlossen - und wenn Erkennen zum Verstehen und Verstehen zum Denken führt, so ist die Idee einer "denkenden Maschine" doch nicht mehr allzu seltsam.

1.6 Künstliche Intelligenz

Dieses Kapitel wäre unvollständig, wollten wir nicht versuchen, auf der Grundlage der vorangegangenen Begriffsklärungen nun zu einer Schlußfolgerung zu gelangen, die zu klären hätte, was es eigentlich mit dem Begriff der Künstlichen Intelligenz auf sich hat.

Sind - im Lichte der vorangegangenen Ausführungen - Maschinen, die geistige, bislang dem Menschen vorbehaltene Funktionen übernehmen, intelligent ?

Und wenn sie in diesem Sinn intelligent sein sollten, reichen dann die Kapazitäten der heute zur Verfügung stehenden Rechner aus, um alle notwendigen denkerischen Funktionen ausführen zu können ?

Sicherlich ist es unrealistisch, zu erwarten - und deshalb soll dies gleich zu Beginn dieses Abschnitts betont werden - , daß kleine Home-Computer , wie zum Beispiel der Commodore C64, "intelligente" Aufgaben übernehmen können, wenn selbst die größten derzeit zur Verfügung stehenden Rechner gerade eben die Oberfläche des ganzen Problembereichs "ankratzen" können (HARTNELL,T. : 1984, S.1).

Es gibt jedoch zwei Bereiche, in denen am ehesten von "intelligenten" Computern gesprochen werden kann : Der Bereich des Lernens und der Bereich des Treffens von Entscheidungen (ebenda, S.2). Ein dritter Bereich, der - historisch gesehen - sogar an erster Stelle genannt werden kann, soll hinzugefügt werden :

Schon "zu Beginn der 60-er Jahre (entwickelten

sich) Forschungsrichtungen mit stark "spielerischer" Note, bei denen versucht wurde, Computer mit "intelligentem Verhalten" auf bestimmten eingeschränkten Gebieten zu versehen. Hiezu gehören die verschiedensten Arten von Spielprogrammen, wie Schach, Dame ... , Backgammon ..., sowie Versuche, den Computer Gedichte oder Kompositionen ... schreiben zu lassen." (KOBSA,A. : 1984, S.108).

Inzwischen gibt es "tatsächlich Hunderte von Spielen, die Computer bewältigen - mit wachsender Geschicklichkeit noch dazu .... (sie stellen zum Teil) "eine echte Herausforderung für die menschlichen intellektuellen Fähigkeiten dar ... (sie demonstrieren) das wachsende intellektuelle Leistungsvermögen künstlicher Systeme..." (SIMONS,G. : 1984, S.93).

Natürlich sind es aber nicht nur die Spielcomputer oder Spielprogramme, die uns unter dem Stichwort Künstliche Intelligenz allein interessieren. Vielmehr geht es um solche Maschinen und Systeme, die diejenigen Probleme bewältigen können, über die oben gesprochen wurde (beispielsweise die Spracherkennung). Solche Computer sollen zum Beispiel in der Lage sein,

- Sprachen sofort automatisch zu übersetzen,
- gesprochene Worte in Schrift umzusetzen,
- sachkundige Auskünfte zu geben,
- Schlüsse zu ziehen,
- Entscheidungen zu treffen u.ä.

Die Arbeiten an der Entwicklung der Computer der sog. 5.Generation verfolgen vor allem diese und ähnliche Zielsetzungen.

Die Forschungen im Bereich der Künstlichen Intelligenz erstrecken sich aber nicht nur darauf, solche Maschinen zu entwickeln, sondern dienen gleichzeitig der Erforschung der Bedingungen von Denkvorgängen ganz allgemein - dies entspricht unseren Überlegungen in einem der vorangegangenen Abschnitte, wo auf den wechselseitigen Charakter

der Beeinflussungen zwischen Mensch und Maschine hingewiesen wurde.

Wenn aufgrund solcher Forschungen Maschinen gebaut werden können, die sich so verhalten, als seien sie intelligent, dann sind sie intelligent - darauf wurde schon hingewiesen. Schon sehr früh haben Wissenschaftler auf diesen Sachverhalt aufmerksam gemacht. So zum Beispiel der berühmte A.TURING, der schon 1947 zu dieser Frage einen bemerkenswerten Aufsatz veröffentlichte :

"Der Titel lautete : "Intelligente Maschinen". In diesem Aufsatz kam TURING zu dem Schluß, daß Maschinen denken können ... TURING erläuterte seine Vorstellungen am Beispiel des menschlichen Säuglings, dessen geistiger Apparat größtenteils unprogrammiert ist. Der Säugling hat viel zu lernen und zu vertehen, ehe er seinen Platz in der Welt einnehmen kann. Deshalb erziehen - das heißt programmieren - wir die kleine Denkmaschine für einige Jahre." (RITCHIE,D. : 1984, S.125).

Auf der Grundlage dieser Überlegungen entstand der sog. TURING-Test , der von folgender Behauptung ausgeht :

"Wenn eine Maschine Sie davon überzeugen kann, daß sie ein intelligentes Verhalten zeigt, dann ist ihr Verhalten auch tatsächlich intelligent." (ebenda, S.128).

In dem umfangreichen Buch von POPPER/ECCLES finden wir hingegen auf die Frage, ob man eines Tages Maschinen bauen kann, welche denken können, die folgende Antwort : "Ich würde ohne Zögern sagen, daß wir es nicht können, ungeachtet meines grenzenlosen Respekts für A.M.TURING, der das Gegenteil meinte. Wir können vielleicht einem Schimpansen das Sprechen beibringen ... Vielleicht können wir eines Tages auch einen Mikroorganismus schaffen, der sich in einer entsprechend aufbereiteten Umwelt aus Enzymen selbst reproduzieren kann ... Aber ich sage voraus, daß

es uns nicht gelingen wird, elektronische Computer mit bewußter subjektiver Erlebnisfähigkeit zu bauen." (POPPER,K.R./ECCLES,J.C. : 1982, S.256).

Wenn wir weiter oben Intelligenz definiert haben als "die Fähigkeit, sich unter zweckmäßiger Verfügung über Denkmittel auf neue Forderungen einzustellen" (STEINACKER,I. : 1984, S.8), dann läßt sich diese Definition sehr gut auf Maschinen, zum Beispiel auf Roboter aber auch speziell auf Computer übertragen :

Solche Maschinen müssen - wenn diese Definition anwendbar sein soll - für einen der folgenden Sachverhalte oder für mehrere von ihnen zuständig sein :

- Wahrnehmen,
- Probleme lösen,
- Schlüsse ziehen,
- Spielen,
- Lernen,
- Wissen anwenden,
- Auskünfte geben,
- Tätigkeiten ausführen.

Diese Aufzählung weist auf eines der wichtigsten Probleme der Forschungen im Bereich der Künstlichen Intelligenz hin, vor dem wir derzeit stehen :

"Das wissenschaftliche Arbeiten in diesem Bereich (geht) noch ziemlich orientierungslos vonstatten ... Die Disziplin befindet sich ... derzeit sicher in keiner besseren Position als die mittelalterliche Alchimie : Kaum theoriengeleitet mischt sie verschiedene Ingredienzen zusammen und hofft, daß dabei so etwas wie "Intelligenz" herauskommt. Dieses "blinde Herumtappen" ist allerdings der Beginn fast jeder neuen Disziplin; auf Basis der experimentellen Befunde der Alchimie konnte die moderne empirische Chemie enstehen, die jetzt ein weitaus stärker theoriengeleitetes Vorgehen ermöglicht." (KOBSA,A. : 1984, S.117).

Es braucht also nicht zu verwundern, daß an dieser Forschungsrichtung - und damit generell auch am Begriff der Künstlichen Intelligenz und an der Idee der "denkenden" Maschinen - sehr ernst zu nehmende Kritik geübt wird. Wir haben ja weiter oben schon einen sehr gravierenden Einwand vorgetragen (POPPER,K.R./ECCLES,J.C. : 1978, S.256).

Diese Kritik setzt insbesondere an der Tatsache an, daß nicht alles, was Maschinen zu tun in der Lage sind, überhaupt noch verstanden wird - insoweit ist nicht mehr in jedem Einzelfall nachprüfbar, ob eine bestimmte Maschine tatsächlich das tut, was das menschliche Gehirn tun würde.

"Eine Sache genügend gut zu verstehen, um in der Lage zu sein, sie für einen Computer zu programmieren, bedeutet nicht, sie bis in ihre tiefsten Tiefen zu verstehen. In der Praxis kann es dieses letzte Verständnis gar nicht geben." (WEIZENBAUM,J. : 1978, S.150). ... "Daß ... Computersysteme ... seit langem das Verständnis derjenigen übersteigen, die mit ihnen arbeiten ... ist eine sehr ernste Entwicklung." (ebenda, S.311).

Allerdings gelangt auch ein Mahner, wie Joseph WEIZENBAUM zu dem folgenden Schluß :

"Ich akzeptiere den Gedanken, daß ein modernes Computersystem hinreichend komplex und autonom ist, daß seine Bezeichnung als Organismus gerechtfertigt ist. Da er seine Umwelt sowohl sensorisch erfahren als auch beeinflussen kann, gestehe ich sogar zu, daß er in einem extrem eingeschränkten Sinne "sozialisiert" werden, d.h. durch seine Erfahrungen mit der Welt modifiziert werden kann. Ich gestehe ferner zu, daß man einen entsprechend konstruierten Roboter bauen kann, der sich gewissermaßen selbst erfahren kann, daß er beispielsweise zu lernen vermag, zwischen seinen eigenen Bauteilen und solchen Gegenständen zu unterscheiden, die sich außerhalb seiner selbst

befinden, daß man ihn dazu bringen kann, in erster Linie seine eigenen Bauteile gegen eine Beschädigung zu schützen und dann erst Objekte, die ihm äußerlich sind, und daß er ein Modell von sich bauen kann, das man in gewissem Sinne als eine Art Selbstbewußtsein ansehen könnte. Wenn ich deshalb sage, daß ich bereit bin, einen solchen Roboter als "Organismus" zu bezeichnen, so erkläre ich mich damit bereit, ihn als eine Art Tier anzusehen. Und ich habe bereits eingeräumt, daß ich keine Möglichkeit sehe, bezüglich des Intelligenzgrades eine Obergrenze zu ziehen, den ein derartiger Organismus zumindest im Prinzip erreichen könnte." (WEIZENBAUM,J. : 1978, S.278).

"Wir kommen aber immerhin zu dem Schluß, daß Computer jetzt oder künftig noch so viel Intelligenz erwerben können, daß ihre Intelligenz jedoch stets gegenüber menschlichen Problemen und Anliegen fremd sein muß." (ebenda, S.282).

Diese Aussage verdeutlicht, daß zumindest in den Bereichen, in denen der Computer menschliche Probleme zu erfassen vermag, sein Einsatz zur Erledigung bestimmter Aufgaben angezeigt ist, um damit menschliche Arbeitskraft zu ersetzen und einzusparen. Auf die sozialen, ökonomischen und politischen Konsequenzen solcher Ersetzungen soll hier nicht eingegangen werden.

Ein breites Spektrum unterschiedlicher Maschinen ist hier vorstellbar : Von den manuelle Tätigkeiten ausführenden Industrierobotern, über Textverarbeitungs- und Übersetzungsautomaten bis hin zu den entscheidungsfindenden Expertensystemen und den sich selbst steuernden Atomraketen.

Diese Beispiele können aber immer noch nicht die Frage beantworten, ob wir solche Maschinen als intelligent bezeichnen dürfen. "Wenn wir glauben, er (der Computer) habe Intelligenz und Macht, bekommt er beides !" (WEIZENBAUM,J. : 1984, S.98).

Diese Warnung des engagierten Kritikers WEIZENBAUM sollten wir im Gedächtnis behalten, wenn wir in

einem Versuch, die vorangegangenen Ausführungen zusammenzufassen, den Begriff der Künstlichen Intelligenz folgendermaßen definieren :

"Künstliche-Intelligenz-Forschung versteht sich als diejenige Disziplin, die versucht, "Computer Aufgaben vollbringen zu lassen, die, wenn sie der Mensch vollführte, Intelligenz verlangen würden" (MINSKY 1968, p.V)." (KOBSA,A. : 1984, S.102/103).

Zu dieser Definition bemerkt Neill GRAHAM :

"Der Grund für die Bedingung "würden sie (die Aufgaben) von einem Menschen erfüllt" ist folgender : Wenn ein Computer dazu programmiert worden ist, etwas bestimmtes zu leisten, kommt man leicht in Versuchung abzustreiten, daß eine solche Leistung tatsächlich Intelligenz erfordert, zumindest bezüglich der Art und Weise, wie der Computer seine Arbeit tut. Es entwickeln sich endlose Argumentketten darüber, welche Handlungen des Computers als intelligent anzusehen sind und welche nicht. Die Mehrzahl dieser Argumente wird nun dadurch gegenstandslos, daß man sagt : Das, was ein solches Programm leistet, würde bei einem Menschen, der dasselbe zu leisten hätte, Intelligenz voraussetzen. Ob der Computer, der eben diese Leistung vollbringt, nun tatsächlich "intelligent" zu nennen ist, ist nicht von Bedeutung und bleibt jedem selbst überlassen." (GRAHAM,N. : 1983, S.9).

Wir schliessen uns diesen Ausführungen an und wollen damit die Erörterung der wesentlichen Grundbegriffe beschliessen.

Kapitel 2 : Ein Blick in die Vergangenheit

2.1 Vorbemerkung

Will man verstehen, welches die Gründe für die große Bedeutung der Computer heute sind, und warum man überhaupt über die Stichworte

- künstliche Intelligenz
- denkende Maschinen

nachdenkt, dann ist es ganz sinnvoll, einen Blick auf die Geschichte der Computerentwicklung zu werfen.

Wir wollen uns dabei auf eine Skizzierung grundlegender Entwicklungstendenzen beschränken, gehen also nicht auf alle Details ein, obgleich auch diese sehr wohl interessant für die Gesamtentwicklung und überdies für sie notwendig gewesen sind. So mancher berühmte Name bleibt deshalb im folgenden unerwähnt und sehr viele bemerkenswerte Erfindungen werden hier nicht beschrieben.

2.2 Die Anfänge

Im vorangegangenen Kapitel wurde schon erwähnt, daß wir Computer als Werkzeuge betrachten können. Es ist deutlich geworden, daß dieses eine etwas einschränkende Betrachtung ist, die eine Reihe wichtiger Aspekte nicht berühren und nicht berücksichtigen kann - allerdings ist diese Betrachtungsweise recht praktikabel, weil sie eines sehr klar stellt :

Die Ursprünge der Entwicklung dieser Geräte kann mit dem Wunsch begründet werden, den Menschen von der Last geistiger Tätigkeiten wenigstens partiell zu befreien.

Schon sehr früh in der Geschichte der Menschheit hat man sich geeigneter Geräte bedient, um körperliche Arbeiten zu erledigen oder sie zumindest zu erleichtern; und deshalb hat man auch die Frage gestellt, ob so nicht auch mit geistigen Arbeiten verfahren werden kann.

Es liegt der Gedanke nahe, daß es sich dabei nicht um solche geistige Tätigkeiten handeln kann, wie zum Beispiel die Niederschrift eines Romans von TOLSTOI, eines Gedichts von HÖLDERLIN oder eines Streichquartetts von SCHUBERT. Aber vielleicht geht es mit einfacheren Aufgabenstellungen.

Einfachere Aufgabenstellungen aus dem Bereich geistiger Arbeiten führen in drei Teilbereiche :

1. Spiele
2. Logik
3. Arithmetik

Beispielsweise bestehen einfache Brettspiele aus einer eng beschränkten Zahl von Spielzügen und wenigen Reaktionsmöglichkeiten des jeweiligen Spielgegners.

Man kann sich sicher vorstellen, daß einer der Spieler durch einen Computer ersetzt werden kann, wenn dieser über ein adäquates Reaktionsprogramm verfügt. Es ist im übrigen auch möglich, daß ein derartiges Spiel von zwei Computern gegeneinander gespielt wird.

Was für einfache Brettspiele gilt, gilt heute dank der gestiegenen Leistungsfähigkeit der Rechner auch für anspruchsvollere Spiele bis hin zum höchst komplizierten Schachspiel . In der Tat gibt es heute Computer (genauer : Programme für Computer), die ganz hervorragend Schach spielen können.

Schon vor einigen Jahrhunderten waren deshalb die Zuschauer fasziniert, wenn sie sahen, wie ein guter Schachspieler im Kampf mit einem schachspielenden mechanischen Automaten verlor. Erst nachdem dieser Automat in ganz Europa für großes Aufsehen gesorgt hatte, stellte sich heraus, daß man einem Betrüger aufgesessen war : In dem Automaten steckte ein Zwerg, der ein guter Schachspieler war. In den modernen Rechnern allerdings - da können wir ganz sicher sein - stecken keine kleinen Betrüger drin.

Der zweite Bereich, der oben genannt wurde, ist der der Logik . Schon im 13. Jahrhundert war ein erfinderischer Kopf damit beschäftigt, eine Maschine zu entwerfen, die logische Schlüsse ziehen sollte. Es handelte sich um den spanischen Mystiker LULLUS, der im Jahre 1275 versuchte, diese Maschine zu entwickeln. Seine Zielsetzung dabei war, mittels einer logisch arbeitenden Maschine die Lehrsätze des christlichen Glaubens beweisen zu können.

Auch in den Jahrhunderten danach wurde immer wieder der Gedanke verfolgt, eine Maschine zu entwerfen, die fähig sein sollte, logische Schlüsse zu ziehen. Vor allem Theologen und Philosophen griffen diese Idee immer wieder auf, so daß das Stichwort von der "denkenden Maschine" letztlich auf sie zurückgeführt werden kann.

Wenn auch die Spielmaschinen immer faszinierten und die "Logikmaschine" große Geister bewegte, so ist doch der wichtigere Impuls in dieser Entwicklung aus dem dritten genannten Bereich gekommen, aus dem Bereich der Arithmetik :

Einfache Rechenprozesse sind geistige Aktivitäten, die sich für eine Automatisierung anbieten : Die Addition der Zahlen 5 und 3 zum Beispiel bedeutet, daß ein Zählprozeß in Gang kommen muß, denn Addition ist ein mehrfaches Zählen. Da die Multiplikation nur eine mehrfache Addition ist, läßt sich auch diese, schon etwas kompliziertere Rechenoperation auf Zählprozesse zurückführen - und dies gilt für andere Rechenaufgaben auch.

Die Automatisierung von arithmetischen Aufgaben hing deshalb in erster Linie davon ab, ob es gelingen würde, Zählprozesse zu automatisieren. Eine Maschine, die zählen kann, ist ein Werkzeug zum Rechnen; sie wird zum Rechner (engl.: Computer).

Der entscheidende Grundbaustein einer solchen Rechenmaschine ist also ein Baustein, der zählen kann, oder informationstheoretisch gesprochen, der Zählimpulse übertragen kann : Benötigt wird ein Zählimpulsübertragungselement .

Die ersten Zählimpulsübertragungsbausteine, die konkret genutzt wurden, waren die Finger der Hände. Zählprozesse, d.h. Additionsaufgaben werden seit jeher und auch heute noch zum Teil mit Hilfe der Finger erledigt. Allerdings stößt man dabei recht schnell an Kapazitätsgrenzen. Man hat deshalb ein Gerät entwickelt, das in der Lage war, mehr als nur zehn Impulse in einem Rechengang zu übertragen, nämlich den sog. Abakus .

Dieses Gerät - ein Holzrahmen mit zehn Drähten, auf die jeweils zehn Kugeln aufgefädelt waren - konnte auch schon größere Rechenaufgaben bewältigen. Die gesteigerte Effizienz dieses Geräts wirkte sich allerdings nur deshalb voll

aus, weil man sich gleichzeitig eines geeigneten Zahlensystems bediente, nämlich des uns geläufigen Dezimalsystems .

Die alten Römer beispielsweise hatten dagegen ein recht umständliches Zahlensystem. Eine Rechnung von der Form

VIII + XIV

ist auch mit der Hilfe eines Abakus kaum sinnvoll durchführbar. Besser geht es, wenn man in das uns geläufige Zahlensystem "umsteigt". Dann lautet die obige Rechnung nämlich

8 + 14

und ist in dieser Form mit einem Abakus gut zu erledigen.

2.3 Mechanische Rechner

Auch Zählimpulsübertragungen mit dem Abakus stoßen rasch an kapazitäre Grenzen : Anspruchsvollere Rechenaufgaben können so nicht erledigt werden und auch die kleineren Aufgaben benötigen unter Umständen viel Zeit, bis die entsprechenden Kügelchen hin- und hergeschoben sind.

Deshalb stand man schon sehr früh vor der Aufgabe, die Zählimpulse auf geschicktere Weise automatisch zu übertragen, d.h. schneller und in größerer Zahl. Als geeignete Instrumente dafür erwiesen sich ineinandergreifende Zahnräder . In solchn zahnradgetriebenen Maschinen konnten Rechnungen auf die Umdrehungen von Zahnrädern zurückgeführt werden.

W.SCHICKARD (1592-1635) baute eine derartige Maschine. Sie war in der Lage, die vier Grundrechenarten auszuführen. In der gleichen Zeit

entwarf auch der berühmte B.PASCAL einen mechanischen Rechner und der große Mathematiker und Philosoph G.W.v.LEIBNIZ entwickelte einen Rechner, der schneller war als alle seine Vorgänger.

Einen der wichtigsten Anstöße erhielt diese Entwicklung im 19. Jahrhundert durch C.BABBAGE. Sein Hauptinteresse galt der Frage, wie man mathematische Tafeln automatisch und vor allem fehlerfrei erstellen könne. Er entwarf dazu eine Maschine, die die notwendigen Rechenschritte in Teilschritte zerlegte - in ein Programm gewissermaßen, welches von der Maschine Schritt für Schritt abgearbeitet werden mußte. Diese Programmierung einer arithmetischen Aufgabe ist eine der wesentlichen Wurzeln der Rechner seither und damit auch der heutigen Computer.

Eine andere Wurzel unserer heutigen Rechner entwickelte sich aus einer ganz andersartigen Aufgabenstellung heraus : Beipielsweise hatte man schon im 18. Jahrhundert damit begonnen, die Herstellung von Webmustern, die sich ständig wiederholten, zu automatisieren. Berühmt geworden ist der Webstuhl des Franzosen JACQUARD aus dem Jahre 1808. Dieser Webstuhl wurde über Lochkarten gesteuert, auf denen das zu webende Muster als Lochkombination vorgegeben war. "Durch die Lochkartenlöcher fielen Drahthaken, die nun die Fäden griffen und den Arbeitsvorgängen zuleiteten. Lochkarten und zugeordnete Maschinen taten, was bislang von Menschenhand gewebt wurde." (WOLTERS,M. : Reinbek, 1974, S.810).

Hier war also eine erste Form der Prozeßsteuerung verwirklicht, so wie BABBAGE die Programmsteuerung zur Lösung arithmetischer Aufgaben realisiert hatte. Diese beiden Wurzeln - mechanische Rechner und lochkartengesteuerte mechanische Geräte - erwiesen sich letztlich jedoch als Sackgasse, wenn man speziell die Entwicklung von Computern vor Augen hat :

Meachanische Rechner konnten nicht sehr groß

gebaut werden, d.h. ihre Speicherkapazitäten waren sehr beschränkt, und natürlich waren keine besonders hohen Rechengeschwindigkeiten zu erzielen, da mechanische Zählimpulsübertragung viel Zeit kostet.

Ein Ausweg aus dieser Sackgasse hing davon ab, ob man leistungsfähigere Bausteine der Zählimpulsübertragung finden konnte.

2.4 Von HOLLERITH bis ZUSE

Der Ausweg aus der Sackgasse wurde die im 19. Jahrhundert aufblühende Elektrotechnik . Eine Reihe von Geräten, die auf dieser Grundlage entwickelt wurden - insbesondere jene Geräte, die der Informationsübertragung dienten - bereiteten die weitere Entwicklung vor, so zum Beispiel der Telegraph, der Morseapparat usw.

Der entscheidende Anstoß kam von einem Sohn pfälzischer Eltern, dem 1860 in Buffalo geborenen H.HOLLERITH. HOLLERITH war mitbeteiligt an der Vorbereitung der amerikanischen Volkszählung des Jahres 1890. Wie schon bei der Zählung zehn Jahre zuvor, war zu erwarten, daß die Auswertung der in großer Zahl anfallenden statistischen Daten sehr viel Zeit beanspruchen würde. Hauptinteresse galt deshalb in dieser Vorbereitungsphase der Frage, wie man diese statistischen Zählungen und Auswertungen beschleunigen könnte.

H.HOLLERITH hatte die Idee, zu diesem Zweck die Daten auf Lochkarten durch Kombinationen von Löchern und "Nicht-Löchern" zu codieren, um diese Kombinationen dann durch geeignete Maschinen lesen zu lassen. Dies ging so vor sich, daß diese Karten mit maschinell geführten Drahtstiften abgetastet wurden, die immer dann, wenn sie auf ein Loch trafen, einen Stromkreis schlossen.

Auf diese Weise gelang es, die einzelnen Zählimpulse sehr viel rascher zu übertragen, als dies per Hand oder mechanisch möglich gewesen war. Der Zählvorgang war automatisiert und die Ergebnisse der Volkszählung von 1890 lagen in der Tat sehr rasch vor.

Diese Idee HOLLERITHs, zu verarbeitende Daten auf Lochkarten zu codieren und die Zählimpulse durch Schließen von Stromkreisen zu übertragen, hatte bis in die Mitte unseres Jahrhunderts hinein Bestand. Wer noch 1950 erklärte, sich zum Hollerith-Fachmann ausbilden zu lassen, erwarb sich Ansehen bei den Zuhörern und galt gewissermassen als "gemachter Mann". Die Hollerith-Maschinen waren längst nicht mehr nur reine Auszählgeräte, sondern sie konnten auch sortieren, sie konnten addieren und 1939 wurde in Deutschland sogar eine Hollerith-Maschine eingesetzt, bei der kurze Rechenprogramme über eine Stecktafel gesteuert werden konnten.

Die Erfindung HOLLERITHs führte - zusammen mit dem Konzept von C.BABBAGE - zu einer lawinenartigen Entwicklung in der Folgezeit. Immer wurde dabei aber die mit diesen beiden Grundideen festgelegten Grundprinzipien beibehalten :

1. Programmsteuerung des Rechners
2. Binäre Informationsdarstellung (Stromkreise offen - geschlossen)

Auf diesen beiden Grundprinzipien aufbauend entwickelte im Jahre 1932 der damals 22-jährige K.ZUSE in Berlin den ersten programmgesteuerten Rechner der Welt, den Rechner Z1.

Z1 war noch ein mechanischer Rechner, aber bei den Nachfolgemodellen verwendete ZUSE als Bausteine zur Informationsübertragung Relais und nutzte so die gegebenen Möglichkeiten der modernen Informationstechnologie seiner Zeit. Der Rechner Z3, der 1941 in Dienst gestellt wurde, arbeitete mit 2600 Relaisschaltungen.

Im Zeitraum weniger Jahre entstanden auch in anderen Ländern, insbesondere in den USA, voll funktionsfähige, programmgesteuerte Rechner auf Relais-Basis.

2.5 "Elektronengehirne"

Das Relais als Impulsübertragungsbaustein wies einen wesentlichen Nachteil auf : es war zu langsam. Immerhin handelt es sich noch um einen halb-mechanischen Baustein, d.h. die einzelne Impulsübertragung kann nicht beliebig beschleunigt werden, so daß die Rechenoperationen insgesamt doch viel Zeit verbrauchen.

Entscheidende Fortschritte waren deshalb davon abhängig, die Übertragungsgeschwindigkeit des Einzelimpulses zu erhöhen.

Das dafür geeignete Instrument war die Elektronenröhre , die als Baustein zur Impulsübertragung quasi mit Lichtgeschwindigkeit arbeitet und somit zu einer Verkürzung der Gesamtrechenprozesse um den Faktor 1000 beitrug. Eine Aufgabe, deren Bewältigung vorher fünf Minuten dauerte, war nun im dritten Teil einer Sekunde zu erledigen. Auf diese Weise sind Rechenzeiten möglich geworden, die die praktische Nutzung der Rechner für die Lösung konkreter Problemstellungen ermöglichte.

Ein erster dieser Elektronenrechner war ENIAC, ein Rechner, der mit ca. 18000 Röhren bestückt war. Dieser Rechner war extrem schnell (im Vergleich zu dem, was man bislang gewohnt war), allerdings auch extrem teuer in der Herstellung und, was das Hauptproblem war, extrem anfällig. Elektronenröhren gehen leicht kaputt und deshalb mußte ENIAC (und seine Elektronen-Brüder) meist

repariert werden - aber wenn er einmal rechnete, dann rechnete er ungeheuer schnell.

In dieser Zeit, Ende der vierziger Jahre, kam das Schlagwort vom "Elektronengehirn" auf. In dieser Namensgebung dokumentierte sich zum ersten Mal die Frage, ob solche außerordentlich leistungsfähigen Maschinen vielleicht besser rechnen können als das menschliche Gehirn oder ob sie vielleicht die Arbeit des menschlichen Gehirns insgesamt ersetzen können. Diese Frage wurde nicht ohne Besorgnis gestellt - insbesondere von den Nicht-Fachleuten.

Die Fachleute hingegen beschäftigten sich mit einer anderen Frage, nämlich damit, die zu störanfällige Elektronenröhre durch einen weniger empfindlichen Impulsübertragungsbaustein zu ersetzen. Als geeigneter Nachfolger erwies sich der 1948 erfundene Transistor , der ab 1958 zur Impulsübertragung in Rechnern verwendet wurde.

Der Transistor war weit weniger störanfällig, vergleichsweise billig in der Herstellung und viel kleiner als die Röhre. So wurde es möglich, Rechner zu bauen, die billiger waren - jetzt wurde auch ihre kommerzielle Nutzung interessant - ,die kaum noch wegen Reparaturen ausfielen, und die noch wesentlich schneller waren als die Elektronenrechner, nicht weil die Geschwindigkeit der Übertragung des Einzelimpulses hätte gesteigert werden können, sondern weil wegen der geringen Baugröße der Transistoren die internen Informationsübertragungswege wesentlich verkürzt werden konnten.

2.6 Mikroprozessor-Technik

Zusammengehörige Bauteile - Transistoren, Kondensatoren, Widerstände - wurden in der Folgezeit zu Modulen zusammengepackt (man kennt solche Bauteile zum Beispiel aus moderneren Fernsehgeräten).

Da letzten Endes jeder Prozeß der maschinellen Datenverarbeitung darauf beruht, daß maschinenintern Informationen übertragen werden, kann man solche Module als "Schalter" bezeichnen - Schalter, die gewissermaßen für die internen Weichenstellungen verantwortlich sind. Mit fortschreitender Technologie sind diese Schalter immer komplexer geworden, d.h. sie sind in der Lage, nicht nur zwei Schaltalternativen, wie zum Beispiel ein einfacher Lichtschalter, sondern sehr viel mehr Alternativen zu realisieren.

Man nannte damals derartige Schalter deshalb "integrierte Schaltkreise" (IC) und setzte in den sechziger Jahren alle modernen technologischen Möglichkeiten ein, diese ICs immer weiter zu verkleinern. Dies wurde vor allem durch eine ganz neue Technologie wesentlich beschleunigt, die sich in der zweiten Hälfte der siebziger Jahre durchsetzte : Die Technologie der "Silizium - Chips" . Sie hat es ermöglicht, daß ganz extreme Miniaturisierungen möglich wurden :

Auf einer Grundfläche von 3*3 Millimetern konnten bis zu 14000 Schaltmöglichkeiten untergebracht werden. Dies bedeutet weitere radikale Verkürzung der Übertragungswege, damit weitere Erhöhung der Rechengeschwindigkeiten und daneben weitere Verringerung der Störanfälligkeit und extrem billige Herstellung dieser sog. " Mikrochips ".

Man nennt einen solchen "Mehrfachschalter" einen Mikroprozessor . Derartige Mikroprozessoren sind die Grundbausteine der heutigen Rechner, sowohl

der Großrechner wie auch der so zahlreichen Homecomputer . Und nicht nur da werden jetzt Mikroprozessoren benutzt, sondern sie sind auch zentrale Bausteine von Industrierobotern, von neuen blockierfreien Autobremsen, von Kfz-Zündungsanlagen, von elektronischem Spielzeug, von völlig automatisch gesteuerten Drehbänken, von Zeichenautomaten u.v.a.

Es ist vielleicht nicht falsch, in diesem Zusammenhang von einer neuen technologischen Revolution zu sprechen, die sicherlich genauso gravierende Auswirkungen ökonomischer, gesellschaftlicher und politischer Art haben dürfte, wie zum Beispiel die technologische Revolution der Industrialisierung im 19. Jahrhundert.

2.7 Die Zukunft

Es ist abzusehen, daß die ständige Verkleinerung der Bauteile auch zukünftig vonstatten gehen wird. Dies bedeutet sicherlich weitere Erhöhung der Rechengeschwindigkeiten und weitere Steigerungen der Speicherkapazitäten. In den modernen Forschungslaboratorien insbesondere in Japan und in den USA wird an neuen und viel kleineren Speichermedien, als man sie bisher kannte, gearbeitet. Zusammen mit neueren Rechnerarchitekturen , die vor allem das gleichzeitige Erledigen gleichartiger Teilaufgaben ermöglichen sollen, werden die Computerkapazitäten noch um ein Vielfaches steigen.

Dies aber ist nicht der wichtigste Aspekt zukünftiger Entwicklungen. Entscheidend ist vielmehr der extreme Preisverfall , der sich im Bereich der Mikroprozessoren schon vollzogen und die rasche Verbreitung der Computer, aber insbesondere auch der mikroprozessorgesteuerten

Maschinen, verursacht hat.

Diese Geräte stehen ja längst nicht mehr nur in Großunternehmen, sondern inzwischen auch in Klein- und Mittelbetrieben, in Handwerksbetrieben, bei Anwälten und Ärzten, bei Architekten und Spediteuren, in modernen landwirtschaftlichen Unternehmen, in Forschungseinrichtungen und in Schulen und in zunehmendem Maße auch in privaten Haushalten.

Und diese Geräte dienen jetzt nicht nur dazu, Rechenaufgaben irgendwelcher Art zu bewältigen, sondern sie leisten heute weit mehr : Man denke an die Stichworte Textverarbeitung ,an die Prozeßsteuerung usw., die vielen Computerspiele u.ä. nicht zu vergessen.

Hinzu kommt, daß die Software für diese Geräte ständig erweitert und verbessert wird : Immer mehr und immer anspruchsvollere Computerprogramme stehen für die Rechner zur Verfügung und immer kompliziertere Aufgaben können dank dieser Programme quasi "per Knopfdruck" erledigt werden. Je "intelligenter" diese Software, desto geringer der entsprechende Aufwand für den Benutzer, d.h. desto mehr Zeit bleibt, sich der Bearbeitung solcher Probleme zuzuwenden, für die der Computer nicht geeignet ist.

Weiterhin ist zu berücksichtigen, daß die Mikroprozessortechnologie auch die Grundbausteine für neue Kommunikationstechnologien , d.h. für die Technologie der neuen Medien bereitstellt. Informationsaustausch - sei es zu häuslichen Unterhaltungszwecken oder zum Beispiel aus wirtschaftlichen Gründen - ist für die hochindustrialisierte Gesellschaft eine Notwendigkeit und gewinnt im Zeichen ständig zunehmender Informationsmengen immer mehr an Bedeutung.

Die Technologie zur Bewältigung zunehmender Informationsmengen ist die Technologie der Mikroprozessoren. Sie erlaubt die Bereitstellung

großer Datenbestände an beliebigen Arbeitsplätzen, sie ermöglicht die sehr rasche Kommunikation zwischen verschiedenen verbundenen Arbeitsplätzen, sie ermöglicht die schnelle Auswertung umfangreicher Datenmengen.

Über die verschiedenen Möglichkeiten der neuen Informationstechnologien soll hier nicht weiter gesprochen werden. Das Entscheidende in diesem Zusammenhang ist für uns, daß diese Terchnologie und die Vernetzung von verschiedenen, unter Umständen sehr vielen Rechnern, den Zugriff auf große Informationsmengen ermöglichen. Wir können leicht auf immer mehr Informationen, auf mehr Fakten, auf mehr Wissen zugreifen, als wir in unserem eigenen Gehirn speichern und zur Verfügung halten können.

Auch die Maschinen selbst können bei der raschen Erledigung der ihnen zugewiesenen Aufgaben auf immer größere Informationsreserven zurückgreifen, sie können immer komplexere Programme abarbeiten. Sie werden - wenn man so will - immer klüger und diese Schlußbemerkung führt uns wieder direkt zu unserem zentralen Thema zurück.

Kapitel 3 : Anwendungsbereiche der künstlichen Intelligenz

3.1 Vorbemerkung

In diesem Kapitel wollen wir an das anknüpfen, was im Kapitel 1 über künstliche Intelligenz gesagt wurde. Dort wurde abschließend festgestellt, daß unter künstlicher Intelligenz alle diejenigen maschinellen Aktivitäten verstanden werden können, die, wenn sie von Menschen ausgeführt würden, Intelligenz erforderten. Diese Betrachtung können wir nun wesentlich präzisieren, nachdem wir im vorangegangenen Kapitel einen Blick auf die bisherige Computerentwicklung geworfen haben und somit etwas realitätsbezogener die Möglichkeiten von Rechnern einschätzen können.

3.2 Computer und andere Maschinen

Die Mikroprozessortechnologie hat nicht nur die Computerentwicklung in den letzten Jahren bestimmt, sondern es sind auch andere Geräte auf der Grundlage dieser Technologie entstanden, von denen man sagen kann, daß sie Aufgaben übernehmen, die den Einsatz menschlicher Intelligenz erforderten, würden diese Geräte nicht zur Verfügung stehen.

Man denke beispielsweise an Bremsanlagen bei Autos, die das Blockieren der Räder bei Vollbremsungen verhindern. Man hat Sensoren entwickelt, die in Sekundenbruchteilen die Verzögerungswerte der gebremsten Räder erfassen und schnell genug den Anpreßdruck der Bremse so weit verringern, daß ein Blockieren rechtzeitig verhindert wird.

Entsprechendes gilt für Maschinen, die selbstätig ein Werkstück ergreifen können (ohne sie zu zerdrücken oder sie, wegen eines zu schwachen Zugriffs wieder fallen zu lassen), transportieren, verarbeiten, wieder absetzen oder weiterreichen können - sog. Industrieroboter . Dies sind nun allerdings Geräte, die wieder dem Ersatz körperlicher Tätigkeiten dienen, so daß sie im Rahmen unserer Themenstellung nur peripher interessieren.

Es gibt darüberhinaus aber auch Geräte etwa zur Überwachung und Steuerung von komplzierten Produktionsprozessen, Arbeitsabläufen, von Fließbändern, zur Herstellung von Mischungen, Verpackungen u.dergl. Derartige und ähnliche Überwachungsaufgaben können sehr wohl wieder zu jenen Tätigkeiten gezählt werden, die geistige Aktivitäten des Menschen ersetzen.

Alle diese Geräte werden Roboter genannt. Häufig sind sie beweglich, in verschiedene Richtungen

drehbar, können ihre Positionen verändern, können über geeignete Kameras sehen und erkennen, können beim Ergreifen von Werkstücken die Zugriffskraft auf das jeweilige Werkstück abstimmen usw. usw. Wir stehen hier ganz am Anfang einer Entwicklung, die sicherlich noch viele Überraschungen bieten wird.

"Den Roboter der Zukunft kann man sich ... als spezielles Expertensystem vorstellen. An die Stelle der Komponenten für Dialogführung und Erklärung treten bei einem solchen Roboter Komponenten für Training und multisensorische Beobachtung der Objektwelt. Hier wie dort sind die eigentlichen "Intelligenzkomponenten" eine Wissensbasis und ein Inferenzsystem. Ein Roboter ist - so gesehen - ein vollständig über seine Objektwelt rückgekoppeltes Expertensystem.

Informationstechnisch treten hier zwei Problemkreise in den Vordergrund. Der eine betrifft die Fähigkeit des Lernens . Das bedeutet, daß sich ein Roboter die erfolgreiche Bewältigung bestimmter Situationen merken soll, um bei vergleichbarer Gelegenheit auf derart adaptives Wissen zurückgreifen zu können. Der andere, besonders eigenwillige Problemkreis liegt in der Notwendigkeit, Signalbilder verschiedener Sensorherkunft ... aufeinander abzubilden." (SCHUCHMANN, H.-R., 1984, S.7/8).

Letzten Endes geht es hier also nicht um "Maschinenwesen", wie man sie sich zum Beispiel in der Science-Fiction-Literatur früherer Jahre vorgestellt hat (etwa in den klassischen Romanen von H.G.WELLS), um Wesen, die sich vielleicht im Verhalten, Handeln und vor allem im Aussehen nicht von Menschen unterscheiden (und deshalb in den Romanen häufig für Aufregung und Verwechslungen, manchmal aber auch für Furcht und Schrecken sorgen), sondern um zweckgerichtet konstruierte Werkzeuge (siehe auch Kap.1).

Zu der Koppelung von Maschinen, die manuelle Tätigkeiten übernehmen, mit solchen, die geistige

Aktivitäten unterstützen, bemerkt D.RITCHIE :

"Die Menschen hatten eine Verlängerung ihres Körpers (Roboter-Arme) mit einer Verlängerung ihres Verstandes (programmierbare Computer) zusammengefügt. Ein künstlicher Verstand wurde mit metallenen Gliedmaßen gekoppelt. Zwei Trends in der Evolution der Technologie (und damit in der Evolution der Menschheit) wurden miteinander verknüpft. Das Ergebnis war der denkende Roboter.

Dies war ein unglaublicher Durchbruch in der Welt der Erfindungen. In ähnlicher Weise, wie die Cro-Magnon-Menschen durch Geist und technologische Überlegenheit ihre Neandertaler-Nachbarn in die Vergessenheit und zum Aussterben trieben, sind heute moderne Menschen mit einem Roboter an ihrer Seite auf dem Vormarsch gegen die Menschen ohne Computer. Das Ergebnis mag als eine der größten sozialen Umwälzungen aller Zeiten - oder Katastrophen - in die Geschichte eingehen." (RITCHIE,D. : 1984, S.135).

Weniger dramatisch ausgedrückt heißt das, daß diese Geräte für gravierende ökonomische, soziale und politische Umgestaltungen der modernen Industriegesellschaften sorgen, insbesondere weil sie billiger, leistungsfähiger, und weniger störanfällig sind als Menschen, keinen Urlaub, keine Lohnfortzahlung im Krankheitsfall und keine 35-Stunden-Woche brauchen.

"Während Menschen einen aktiven und phantasievollen Verstand haben, der nach stundenlanger monotoner Arbeit abschweift, kann ein Roboter seinen rudimentären Verstand über endlose Zeiträume auf ein bestimmtes Ziel richten. Dies ist ein entscheidender Grund, warum immer mehr Roboter auf dem Arbeitsmarkt eingesetzt werden. Sie setzen Autos für uns zusammen, backen unser Brot, raffinieren unser Benzin, füllen unsere Getränke in Flaschen ab." (RITCHIE,D. : 1984, S.141).

"...eine in BUSINESS WEEK im Jahre 1982

veröffentlichte Zahl verhieß allein für die USA einen Verlust an Arbeitsplätzen (durch diese Technologie, der Verf.) in der Größenordnung von 22 bis 25 Millionen" (ebenda, S. 147). In der Bundesrepublik Deutschland rechnet man damit, daß im vor uns liegenden Jahrzehnt etwa die Hälfte der Arbeitsplätze in der ein oder anderen Weise durch diese Technologie beeinflußt wird.

Wir wollen uns an der ausufernden Diskussion um die Industrieroboter und um deren zukünftigen Möglichkeiten hier nicht beteiligen, sondern uns auf die vorangegangenen Bemerkungen beschränken. Die folgenden Ausführungen haben auschließlich mit Computern zu tun, auch wenn dies sicherlich eine Einschränkung des generellen Themas "künstliche Intellignz" darstellt. Mit dieser Beschränkung wird aber noch ein sehr großer Teil derjenigen Themenbereiche erfaßt, die üblicherweise gemeint sind, wenn über künstliche Intelligenz gesprochen wird.

Es geht also um die Frage, was es mit diesem Begriff unter dem Gesichtspunkt des konkreten Computereinsatzes auf sich hat - speziell des Kleinrechnereinsatzes, weil wir uns mit diesem Buch ja an den C64-Besitzer wenden.

3.3 "Denkmaschinen"

Welche Aufgabenstellungen werden von Computern bearbeitet ?

Unter dem Oberbegriff "künstliche Intelligenz" sind es eine Reihe ganz typischer Fragestellungen, die heute (oder zukünftig) per Computer bewältigt werden können.

Es gibt verschiedene Möglichkeiten diese sehr unterschiedlichen Fragen der Klarheit wegen zu klassifizieren und je nach Interessenlage desjenigen, der sich mit diesem Thema beschäftigt, werden unterschiedliche Teilgebiete in den Vordergrung der Betrachtung gestellt. Für die folgenden Ausführungen empfiehlt es sich, von einem möglichst breiten Ansatz auszugehen. Deshalb klassifizieren wir wie folgt :

- Theoretische Fragestellungen
- Expertensysteme
- Suchsysteme
- Entscheidungen
- Erkennen
- Spiele
- Selbstlernende Systeme
- "Kunst"

Zur Erläuterung dieser Themenbereiche dienen die folgenden Ausführungen.

3.3.1 Theoretische Fragestellungen

Computer können - wie schon beschrieben wurde - als Simulationsmaschinen verstanden werden, die, ihrerseits Produkte des menschlichen Geistes, gerade diesen nachahmen, also simulieren sollen.

Insoweit ist der Computer das konkrete Modell eines theoretischen Konstrukts : Der Versuch, das menschliche Gehirn "nachzubilden", und damit auch der Versuch, die Funktionsweise des Gehirns zu verstehen, ist eine der Wurzeln der Computerentwicklung.

In der wissenschaftlichen Literatur wird deshalb der Begriff der künstlichen Intelligenz manchmal sogar gleichgesetzt mit der Aufgabenstellung "Erforschung der Wirkungs- und Funktionsweise des menschlichen Gehirns".

Unter diesem Aspekt haben Computer eindeutig modelltheoretisch erklärenden Charakter. Sie sind weniger anwendungsorientierte Werkzeuge, sondern Simulationsinstrumente.

Dieser Aspekt steht unter Praxisgesichtspunkten aber nicht an erster Stelle.

3.3.2 Expertensysteme

Wenn in der wissenschaftlichen Literatur über künstliche Intelligenz gesprochen wird, dann findet man als praktische Anwendungsbeispiele in der Regel immer die sog. Expertensysteme.

Dank der gestiegenen Speicherkapazitäten der Rechner gelingt es heute, so viele Informationen zu speichern und auf Verlangen des interessierten Computerbenutzers bereitzustellen, daß große Datenbanken errichtet werden können.

Solche Datenbanken dienen dazu, zu bestimmten Sachgebieten alle zur Verfügung stehenden Informationen abrufbar bereitzustellen. Kein Mensch, auch nicht der befähigste Experte, kann alles über ein bestimmtes Spezialgebiet wissen. Natürlich sind aber die Antworten auf bestimmte Fragen um so zuverlässiger, je mehr Informationen

gleichzeitig bei der Beantwortung berücksichtigt werden können. Gerade deshalb ist es sinnvoll, sich die Speicherkapazitäten von Rechenanlagen, die ja ständig erweitert worden sind, zunutze zu machen.

"Expertensysteme stellen eine der eindrucksvollsten Entwicklungen der AI (lies : Artificial Intelligence = künstliche Intelligenz, der Verf.) dar. Ein Expertensystem verfügt über eine Wissensbasis, die das spezialisierte Wissen von Experten enthält, und ist in der Lage, sein Wissen für konkrete Fälle anzuwenden. Damit wird Expertenwissen für einen größeren Benutzerkreis verfügbar.

Auf dem Sektor der Medizin gibt es bereits einige Expertensysteme. Das Programm MYCIN identifiziert Bakterien und macht Vorschläge für die Antibiotikatherapie, CADUCAEUS diagnostiziert innere Krankheiten und PUFF interpretiert Lungenfunktionstests.

Das System PROSPECTOR, ein geologisches Expertensystem zum Auffinden von Mineralien mittels Analyse von Gesteinsproben, hat bereits seine Entwicklungskosten eingebracht, indem es ein Molybdänvorkommen entdeckt hat, das geschätzte 100 Mio Dollar wert ist." (STEINACKER,I. : 1984, S.16).

Auch auf dem Gebiet der Rechtswissenschaften gibt es interessante Expertensysteme, in denen Tausende von Gesetzestexten, Verordnungen und Urteilen gesammelt sind und auf Anfrage bereitstehen. So ist es also leicht feststellbar, welche Urteile in letzter Zeit zu einem bestimmten Straftatbestand ergangen sind, welche Präzedenzfälle vorliegen - eine für Juristen sicherlich sehr angenehme Hilfestellung.

"Mit dem Fifth Generation Computer Systems Project wurde aber von Japan aus ein qualitativer Sprung angekündigt : Expertensysteme sollen für viele Bereiche des Lebens, z.B. Fischfang, Autoreparatur

entwickelt werden. Diese Systeme werden dann auf einem voraussichtlich tragbaren Computer (vermutlich japanischer Herkunft) an jedem Ort anzuwenden sein. Da Expertensysteme ja nicht nur aus einer Datenbasis, sondern auch u.a. aus den für ein Gebiet spezifischen Schlußfolgerungen bestehen ..., bedeutet dies, daß nicht nur Information, sondern auch Wissen kaufbar sein wird ("Knowledge Economics"). Wissen wird dann als Ware in manchen Bereichen Materie, Energie und Information verdrängen.

Dies bedeutet aber für jene Personen, bzw. jene Länder, die sich solche AI-Produkte finanziell leisten können, einen enormen Vorsprung" (TRAPPL,R. : 1984, S.200).

"Das Ziel wissensbasierter Systeme ist es, dem Benutzer das Wissen und die Performanz eines Experten zur Verfügung zu stellen. Dadurch, daß der Computer einen Experten simuliert, wird hochspezialisiertes Wissen, das früher nur über (kostspielige) Experten zugänglich war, allgemein verfügbar. Außerdem kann mit wissensbasierten Systemen die Verbreitung neuer wissenschaftlicher Erkenntnisse und Forschungsergebnisse gefördert werden ... Beide Entwicklungsrichtungen schließen einander nicht aus..." (RETTI,J.: 1984, S.75).

Einen Überblick über Expertensysteme, die zur Verfügung stehen, finden wir bei HORN (HORN,W. : 1983). Die wichtigsten Einsatzgebiete dieser Expertensysteme sind die folgenden :

- Medizin
- Chemie
- Molekulargenetik
- Geologie
- Sprachverstehen
- Mathematik
- Design und Planung
- Militärische Anwendungen

Details dazu finden sich zum Beispiel bei RETTI (RETTI,J. : 1984, S. 88-92).

3.3.3 Suchsysteme

Bei dem Themenbereich, der hier als Suchsysteme bezeichnet wird, handelt es sich um Aufgaben, die eng mit dem im vorangegangenen Abschnitt angesprochenen Problembereich zusammenhängen.

Insbesondere dann, wenn größere Informationsmengen bereitgestellt werden, ist es außerordentlich wichtig, daß "intelligente" Suchalgorithmen zur Verfügung stehen. Nur so kann in akzeptabler Zeit der Informationsvorrat unter den interessierenden Gesichtspunkten durchsucht werden. Sehr große Datenbanken wären fast wertlos, wenn nicht mit geeigneten Suchalgorithmen rasch auf die Daten zugegriffen werden könnte. Deshalb weisen wir diesen Problembereich als eigene Fragestellung gesondert aus, obwohl auch vertretbar wäre, sie bei der Besprechung von Expertensystemen zu behandeln.

Im Prinzip geht es hier darum, einen Computer optimale Suchprozesse ausführen zu lassen. Das Optimalitätskriterium ist dabei üblicherweise die Zeit, die benötigt wird, um ganz bestimmte Informationen zu finden und auszugeben. Es müssen dabei in der Regel sehr unterschiedliche Wege miteinander verglichen werden, um festzustellen, welches der jeweils sinnvollste Suchweg ist.

"Bei vielen Problemstellungen ist es möglich, aufgabenspezifische Informationen in den Suchprozeß aufzunehmen und dadurch den Suchaufwand zu reduzieren. Solche aufgabenspezifische Information nennt man heuristische Information ; die Suchprozesse, die solche Information verwenden, heuristische Suche." (HORN,W. : 1984, S.32).

Anwendungsbeispiele für derartige Suchalgorithmen

finden sich nicht nur im Zusammenhang mit dem Zugriff auf Datenbanken bei Expertensystemen, sondern beispielsweise auch in "intelligenten" Spielprogrammen; man denke etwa an das Schachspiel.

Gibt der Benutzer des Schachprogramms einen Zug vor, so muß der Rechner alle Antwortmöglichkeiten überprüfen, die eventuellen Reaktionen des Benutzers auf jede dieser möglichen Antworten, die Antworten auf jede Reaktion des Rechners usw. usw. Auf diese Weise entstehen sehr viele durchaus unterschiedliche Pfade , die der Rechner zu überprüfen hat. Die Auswahl des optimalen Pfades (des Pfades also, der zum Spielgewinn führt) ist ein Suchproblem, das nicht beliebig viel Zeit beanspruchen darf.

Gute Schachprogramme sind in der Lage, bis zu sechs Züge im Voraus zu kalkulieren - sie betrachten also nur den näherliegenden Teil der jeweiligen alternativen Pfade - die kompletten Pfade können auch sie (noch) nicht bewältigen, dazu reichen die Rechenkapazitäten nicht aus.

3.3.4 Entscheidungen

Einer der interessantesten Anwendungsbereiche der Programme der künstlichen Intelligenz hat mit zu treffenden Entscheidungen zu tun. Es geht präziser gesprochen eigentlich darum, "Entscheidungen unter Ungewißheit" zu treffen, also beispielsweise zwischen zwei gegebenen Handlungsalternativen zu entscheiden, wenn beide Alternativen mit Risiken behaftet sind.

Wenn also, wie immer man entscheidet, das Risiko der Fehlentscheidung besteht, dann stellt sich in der Tat die Frage, ob leistungsfähige Computer mit den riesigen Mengen von Informationen, auf die sie zurückgreifen können, nicht besser in der Lage sind, rascher die Vor- und Nachteile verschiedener

Alternativen gegeneinander aufzurechnen, als der Mensch dies könnte.

Auf jeden Fall ist sicherlich demjenigen, der Entscheidungen zu treffen hat, damit gedient, wenn von einem Computer die für ihn relevanten und die irrelevanten Alternativen voneinander getrennt und ausgegeben werden, wenn die Vor- und die Nachteile der einzelnen Alternativen quantifiziert und insbesondere wenn die Risikowahrscheinlichkeiten berechnet werden können.

Vor allem dieser letzte Aspekt, die Berechnung der Wahrscheinlichkeiten von Fehlentscheidungen , ist außerordentlich wichtig und die Ergebnisse sind von großem informativen Wert. Mit Hilfe der Wahrscheinlichkeitsrechnung können geeignete Computerprogramme diese Aufgaben rasch erledigen.

Man muß sich darüber im klaren sein, daß das Treffen risikobehafteter Entscheidungen nicht nur eine Sache von Managern und Unternehmern ist. Entscheidungen fallen fortwährend auch in anderen Bereichen an. Typisches Beispiel dafür ist das Herbeiführen einer Entscheidung darüber, ob eine formulierte Hypothese (eine Aussage über irgendwelche Sachverhalte, über Tatbestände, über Zusammenhänge oder Entwicklungen) im Lichte empirischer Befunde als bestätigt gelten kann, oder ob sie zu verwerfen ist. Diese Strategie ist einer der zentralen Bausteine jeder empirischen Wissenschaft , also jeder Wissenschaft, die auf (sinnlich wahrnehmbaren) Erfahrungen aufbaut.

Entscheidungen trifft aber auch der Gymnasiast, der in der Oberstufe Kurse und Fächer wählen muß; Entscheidungen trifft die Hausfrau, wenn sie über die Aufteilung des Haushaltsgeldes auf alternative Güter nachdenkt; Entscheidungen trifft jeder, wenn er morgens beim Verlassen des Hauses überlegt, ob er einen Schirm mitnehmen soll oder nicht.

Nicht alle diese Fragestellungen sind so wichtig, daß der Computereinsatz sinnvoll erscheint. Bei manchen Fragestellungen sind aber die Konsequenzen

der jeweiligen Entscheidungen unter Umständen so gravierend, daß ein Computer, der nun also die Entscheidungsfindung durch geeignete Informationen unterstützt oder sogar selbst herbeiführt, ohne Zweifel nützliche Dienste leistet.

Die folgenden Beispiele illustrieren solche Fragestellungen :

Erhöht ein neues Medikament die Heilungschancen bei einer bestimmten Krankheit ?

Sind bestimmte Materialien bruchfester und stabiler als die bisher benutzten ?

Rentiert sich eine geplante Werbekampagne für ein neues Produkt ?

Die Bedeutung solcher und ähnlich gelagerter Fragestellungen wird vielleicht manchem Leser dadurch einsichtiger, daß man darauf hinweist, daß vor allem auch Militärs und Strategen häufig, wenn auch hoffentlich nur theoretisch, vor taktischen Alternativen stehen, zwischen denen entschieden werden muß - und das möglichst rasch. Auch in diesem Anwendungsbereich wird der Computer immer unentbehrlicher und ist es eigentlich schon von Anfang seiner Entwicklung an gewesen.

Die ersten funktionstüchtigen Computer, die gebaut wurden - man denke an den legendären ENIAC aus den vierziger Jahren - waren ja schon in der Herstellung so extrem teuer, daß nur wirklich "finanzkräftige Kunden" sich den guten Rat dieser Maschinen einholen konnten - und solche Kunden waren insbesondere und zuerst die Militärs.

Es geht beispielsweise das Gerücht, daß die Entscheidung darüber, ob während des Korea-Kriegs Anfang der fünfziger Jahre die USA Atombomben gegen das angreifende und gerade im Jahr 1949 kommunistisch gewordene China einsetzen sollte oder nicht, durch einen Computer herbeigeführt worden sei.

Ob Gerücht oder nicht sei dahingestellt - denkbar wäre sehr wohl, daß ein Computer eine derartige Entscheidung fällt, vermutlich sogar rationaler kalkuliert, als dies ein ehrgeiziger General oder Oberbefehlshaber könnte.

3.3.5 Erkennen

In der großen Zahl von Veröffentlichungen, die sich mit dem Thema künstliche Intelligenz beschäftigen, taucht immer wieder ein Stichwort an zentraler Stelle auf, das Stichwort "Erkennen".

Adäquate Informationsverarbeitung, angemessene Entscheidungen und Reaktionen, vernünftige Antworten durch eine Maschine sind nur dann zu erwarten, wenn sie die Sachverhalte oder Tatbestände, um die es geht, erkennen kann. Ohne Kenntnis der Anforderungen, die an sie gestellt werden, ohne Erkenntnis also, kann eine Maschine nicht angemessen reagieren und nichts von dem, was wir - wenn überhaupt - als intelligent bezeichnen könnten, würde sich offenbaren.

Das Stichwort "Erkennen" umreißt dabei ein sehr großes Gebiet. Es bezieht sich beispielsweise auf das

- Erkennen graphischer Muster
- Erkennen von Figuren und Strukturen
- Erkennen von akustischen Signalen
- Erkennen von ertasteten Gegebenheiten
- Erkennen der menschlichen Sprache

Diese kurze Aufzählung ist nicht vollständig, umreißt aber die wichtigsten der Aspekte, die hier eine Rolle spielen. Nur wenn Computer sich mit solchen Sachverhalten befassen können, sind sie fähig zum Wissenserwerb ; nur auf dieser Grundlage sind Leistungen möglich, die - würden sie von Menschen erbracht - Intelligenz erforderten.

Typische Einsatzgebiete derartiger erkennender Computerprogramme sind das Sortieren von Gegenständen nach unterschiedlichen Kriterien (Form, Farbe, Gewicht, Oberflächenbeschaffenheit etc.), die Reaktion auf irgendwelche Signale (zum Beispiel bei den sog. Industrierobotern), die Befolgung verbal eingegebener Anweisungen, Übersetzungen u.ä.

Es versteht sich, daß gerade hier - beim Problem des Erkennens von außerhalb des Computers selbst befindlichen Sachverhalten - beträchtliche Schwierigkeiten zu erwarten sind; nicht zuletzt deshalb konzentrierte sich die künstliche Intelligenz-Forschung häufig auf die Bereiche " Sprachverstehen " und/oder " Mustererkennung " und wegen dieser Konzentrierungen sind in diesen Bereichen auch bemerkenswerte Ergebnisse erzielt worden.

3.3.6 Spiele

Wir haben ja schon an anderer Stelle darauf hingewiesen, daß der frühe Versuch, "spielende" Automaten zu bauen, einer der ersten Ansätze war, "intelligente" Maschinen zu entwickeln.

Typisches Beispiel für die heutige Situation in diesem Bereich ist die große Zahl existierender Schachprogramme . Derartige Programme weisen, dank der Speicherkapazitäten der heutigen Rechner, teilweise einen qualitativen Standard auf, der auch schon geübte Schachspieler durchaus in Schwierigkeiten bringen kann. Insoweit spielt so manches Schachprogramm "intelligenter" als viele Hobby-Schachspieler und manchmal auch wesentlich intelligenter als der Erzeuger des Programms selbst, weil jener - wenn er konkret spielt - häufig nicht mehr all diejenigen Spielzüge im Kopf bewältign kann, die er in immer mühevoller und schrittweiser Detailarbeit dem eigenen Programm mitgegeben hat.

Wenn zusätzlich diese Programme so angelegt sind, daß sie aus eigenen Fehlern lernen können, diese also niemals ein zweites Mal begehen, bzw. wenn sie sich offenbar überlegene Spielzüge des Programmbenutzers "abschauen" und sich eigene Erfolge merken können und das so Gelernte nicht wieder vergessen (im Gegensatz zum menschlichen Spieler), dann werden sie immer besser und immer häufiger gewinnen können.

Wenn Computer also in diesem Sinne intelligent Schach spielen können, so gilt dies auch für andere Spiele, so zum Beispiel für Kartenspiele wie "17 und 4", Brettspiele wie "Dame" oder "Superhirn", strategische Spiele wie "Schiffe versenken" u.ä.

Selbst solche Spiele können in die Betrachtung mit aufgenommen werden, bei denen Zufallselemente eine größere Rolle spielen. Typisches Beispiel dafür ist etwa das altbekannte "Mensch ärgere dich nicht". Auch bestimmte Ratespiele könnten in diesem Zusammenhang genannt werden.

3.3.7 Selbstlernende Programme

Der vielleicht faszinierendste Bereich der intelligenten Programme umfaßt jene Ansätze, die man als "selbstlernende Programme" bezeichnet. Hier geht es darum, daß der Computer von sich aus, dank eines geeigneten Programms, während der Abarbeitung dieses Programms "klüger" wird.

Der Rechner lernt im Dialog mit dem Benutzer, indem er beispielsweise dem Benutzer Fragen stellt, durch deren Beantwortung er neue, verwertbare Informationen erhält. Je mehr solcher Informationen er schon gesammelt hat, um so präziser kann er weiterfragen, d.h. um so rascher gelangt er zu neuen, verwertbaren Informationen.

Es versteht sich, daß solche Programme im Rahmen von Expertensystemen gute Dienste leisten können, aber auch bei Suchprogrammen, bei Spielprogrammen und in ähnlich gelagerten Anwendungsbereichen.

Der wesentliche Effekt bei solchen selbstlernenden Programmen besteht darin, daß Informationen, die der Computer erworben hat, ihm ständig zur Verfügung stehen - er vergißt sie nicht wieder . Hinzu kommt, daß er wegen der außerordentlich raschen Zugriffszeiten moderner Rechner, sehr schnell im Bedarfsfall diese Informationen verwerten kann.

Die selbstlernenden Programme sind deshalb sicherlich wesentliches Element zukünftiger Entwicklungen im Bereich der künstlichen Intelligenz.

3.3.8 Kunst

Wir wollen hier einen letzten Bereich anfügen, der in der wissenschaftlichen Literatur zur künstlichen Intelligenz in der Regel zu kurz kommt, teilweise auch garnicht angesprochen wird, bzw. manchmal in dem ein oder anderen der schon genannten Bereiche mit erfaßt wird.

Gemeint ist der Bereich der "Computer-Kunst".

An anderer Stelle wurde schon das Argument vorgetragen, daß ein Computer allein schon deshalb nicht intelligent sein könnte, weil er kein Gedicht schreiben kann. Diese Auffassung muß - wenigstens partiell - hier revidiert werden : Wenn zu dem Bereich der Kunst beispielsweise auch das "Spielen" mit Farben und Formen, mit Worten und Tönen gezählt wird, dann können auch Computer künstlerisch tätig sein.

Es leuchtet allerdings ein, daß in diesem Zusammenhang das Wort "Kunst" in Anführungszeichen

zu setzen ist, weil etwa das zufallsgesteuerte Spiel mit farbigen Elementen, um ein solches Beispiel konkret zu nennen, doch allenfalls nur "Grundlage" darauf aufbauender - formender - künstlerischer Aktivitäten sein dürfte - aber immerhin : Auch diese Grundlage ist, wenn sie durch einen Computer erstellt wird, ein bemerkenswertes Ergebnis.

ZWEITER TEIL:

Der Computer

Kapitel 4 : Zur Funktionsweise moderner Rechner

4.1 Vorbemerkung

Im dritten Teil dieses Buchs werden kapitelweise Programmbeispiele vorgestellt, die illustrieren sollen, was unter "künstlicher Intelligenz" verstanden werden kann - und zwar solche Programmbeispiele, die auf einem der heutzutage handelsüblichen Homecomputer nachvollziehbar sind, nämlich auf dem C64 von Commodore.

Um die Verbindung zwischen den theoretischen Ausführungen des ersten Teils und den gebotenen Illustrationsbeispielen im dritten Teil des Buchs herstellen zu können, ist es sinnvoll, auch einige Hintergrundinformationen über die Funktionsweise eines solchen modernen Kleinrechners zu liefern. Damit wird es für den Leser leichter, die Abläufe zu verstehen, die bei der Abarbeitung solcher Programme eine Rolle spielen.

Wir gehen dabei allerdings nicht so weit, in die Tiefen der Computertechnologie einzusteigen. Es ist nicht so wichtig, zu wissen, wo welche Stromimpulse was bewirken oder in wirklich allen Einzelheiten zu verstehen, wie eine bestimmte Schaltung innerhalb des Computers funktioniert oder was sie bewirkt. Wichtig ist vielmehr eine Vorstellung darüber, wie die verschiedenen Bausteine einer Rechenanlage zusammenwirken, was die wesentlichen auftauchenden Begriffe bedeuten und welches die prinzipiellen Abläufe sind.

Derjenige Leser, der an technischen Details interessiert ist, möge die Spezialliteratur zu Rate ziehen, die im Literaturanhang zu diesen Themenbereichen genannt ist.

4.2 Grundbegriffe

Der Commodore C64, für den hier Programme vorgestellt werden, ist ein Kleincomputer. Er gehört damit zu derjenigen Klasse von Rechnern, die man heute als Homecomputer bezeichnet. In der Funktionsweise eines solchen Homecomputers gibt es keine prinzipiellen Unterschiede zu größeren Rechenanlagen, so daß die folgenden Ausführungen nicht nur auf diesen speziellen Homecomputer zugeschnitten sind.

Um klarzustellen, wie solche Homecomputer in das große Spektrum von Datenverarbeitungsanlagen überhaupt einzuordnen sind, ist es sinnvoll, diese Anlagen in folgende Gruppen einzuteilen :

- Groß-Computer
- Rechner der mittleren Datentechnik
- Kleine Geschäftssysteme
- Tischcomputer
- Lern- und Übungscomputer
- Taschencomputer
- Taschenrechner

(nach : SACHT,H.-J. : 1981, S.13).

Die Homecomputer sind sinnvollerweise zwischen "Tischcomputern" und "Lern- und Übungscomputer" einzuordnen; allerdings sind die Übergänge zu den benachbarten Kategorien fließend.

Versucht man, die Einsatzgebiete solcher Rechenanlagen alle mit einem einzigen Stichwort abzudecken, so gelangt man zum Begriff der "Datenverarbeitung" :

Im weitesten Wortsinn geht es bei allen Rechnereinsätzen darum, Daten in adäquater Weise zu verarbeiten (in den bisherigen wie auch in den folgenden Ausführungen gehen wir immer von einer Gleichsetzung der Begriffe "Computer" und

"Rechner" aus, obwohl wir ja schon in den vorangegangenen Kapiteln erfahren haben, daß Computer viel mehr können als "nur" rechnen).

Dies läßt es sinnvoll erscheinen, hier dem Begriff der "Daten" zentrale Bedeutung einzuräumen :

Die Datenverarbeitung, wie man sie heute versteht und betreibt, faßt den Begriff der Daten sehr weit : Es kann sich dabei um Ziffern oder Zahlen handeln, aber auch um Buchstaben, Worte und Texte und sogar Sonderzeichen irgendwelcher Art. In diesem Sinne kann man alles als Daten bezeichnen, was für uns von informativem Wert ist :

Daten sind Symbole für Informationen .

Um derartige Daten in einen Rechner eingeben zu können, ist es erforderlich, im Rechner für diese Daten Platz zu schaffen. Dies geschieht durch Reservierung von Speicherplätzen , denen, damit der Rechner sie selbst verwalten kann, Namen zuzuordnen sind.

Bei der notwendigen Namensvergabe ist es zweckmäßig, zwei Typen von Speicherplätzen, die man auch Speicherfelder oder kurz Felder nennt, voneinander zu unterscheiden :

- numerische Felder
- String-Felder

Die Werte , die in numerischen Feldern gespeichert werden, müssen Zahlen sein, die in String-Feldern sind beliebige Zeichenketten (Strings), zum Beispiel Worte eines Textes.

Ein Feld ist derjenige Bereich im Rechner, der für den Wert einer Variablen zur Verfügung steht.

Wir wenden uns nun im folgenden Abschnitt dem Begriff der Datenverarbeitung zu.

4.3 Datenverarbeitung

Unter "Datenverarbeitung" faßt man all diejenigen Verfahren zusammen, die dazu dienen, einen gegebenen Datenbestand im Hinblick auf präzise formulierte Fragestellungen auszuwerten.

Die für die jeweils vom Benutzer gewünschten Verarbeitungsprozesse notwendigen Arbeitsschritte müssen - zusätzlich zu den zu verarbeitenden Daten selbst - dem Computer in Form von Anweisungen in einer ihm verständlichen Sprache, d.h. in einer Programmiersprache mitgeteilt werden.

So entsteht eine Folge von Anweisungen, die man Programm nennt.

Der Rechner benötigt also zwei unterschiedliche Typen von Informationen, wenn er für uns eine Aufgabe erledigen soll :

- zu verarbeitende Daten
- Programmanweisungen

Hinzu kommt noch eine dritte Gruppe von wichtigen Informationen, nämlich solche, mit denen der Benutzer Einfluß auf die Organisation des Datenverarbeitungsprozesses nehmen kann. Solche Informationen nennt man Kommandos .

Kommandos sind Mitteilungen an das Betriebssystem des Rechners, wobei man unter "Betriebssystem" alle diejenigen Dienst- und Hilfsprogramme versteht, die (rechnerintern) für die komplette organisatorische Abwicklung von Prozessen der Datenverarbeitung zuständig sind.

Es versteht sich, daß alle Informationen, die wir einem Rechner geben wollen, ihm so mitgeteilt werden müssen, daß er sie auch verstehen kann. Da die modernen Rechner gemäß den Grundprinzipien der

Elektrotechnik funktionieren, heißt dies, daß alle Informationen für den Rechner durch Belegung von Informationseinheiten verständlich werden, die jeweils nur ein Schließen bzw. Offenlassen eines Stromkreises repräsentieren.

Man bezeichnet diese beiden Informationszustände der Einfachheit halber mit

0 und 1

Die Kommunikation mit einem Rechner beruht also letztlich auf dem Binärprinzip .

Dieses Binärprinzip (auch Dualprinzip genannt) kann man sich am leichtesten klar machen, wenn man sich überlegt, daß beispielsweise Zahlen sehr einfach, statt im Dezimalsystem , an das wir uns gewöhnt haben (mit den zehn Ziffern 0,1,2,3,4,5,6,7,8,9), auch in einem Zahlensystem mit nur zwei Ziffern (0 und 1) dargestellt werden können - im sog. Binärsystem .

Dieses Binärsystem benutzt als Stufenzahlen nicht die Werte

1, 10, 100, 1000 usw.

wie das Dezimalsystem, sondern die Werte

1, 2, 4, 8, 16, 32 usw.

Während die Dezimalzahl 254 also bedeutet :

$$254 = 2*100 + 5*10 + 4*1$$

bedeutet zum Beispiel die Binärzahl 10011

$$10011 = 1*16 + 0*8 + 0*4 + 1*2 + 1*1$$

$$= 16 + 0 + 0 + 2 + 1 = 19 \text{ (dez.)}$$

Entsprechend benutzt das für Computer sehr wichtige Hexadezimalsystem die Stufenzahlen

1, 16, 256, 4096 usw.

4.4 Codes

Der Informationsaustausch mit einem Rechner auf dem Weg über das Binärsystem ist für den Benutzer sehr umständlich und zeitraubend. Dank geeigneter Übersetzungsprogramme kann man aber die uns geläufigen Symbole benutzen. Sie werden durch Codes in die maschinenverständliche Form zurückübersetzt.

Ein Code dient dazu, Informationen zu übertragen oder genauer, Informationen so aufzubereiten, daß der jeweilige Empfänger sie tatsächlich empfangen und auch verstehen kann. Dies gilt auch, wenn der Empfänger kein Mensch, sondern beispielsweise eine Maschine ist.

Der Code, mit dem der Commodore C64 arbeitet, ist der sog. ASCII-Code .

In diesem Zusammenhang ist es wichtig, zwei wesentliche Grundbegriffe der Datenverarbeitung kennenzulernen :

Die kleinste Informationseinheit in einem Rechner, die nur 0 oder 1 speichern kann, wird bit genannt.

Für jedes Symbol, das im Rechner dargestellt

werden soll, wird eine Folge von 8 bits reserviert. Diese Folge wird Byte genannt.

Beispielsweise stellt sich der Buchstabe A gemäß des ASCII-Codes als folgende Byte-Besetzung dar :

A = 01000001

Die Speicherkapazität eines Rechners bemißt sich nach der Anzahl von Symbolen, die gleichzeitig gespeichert werden können, also nach der Anzahl von Bytes, die für den Benutzer zur Verfügung stehen. Dabei gilt :

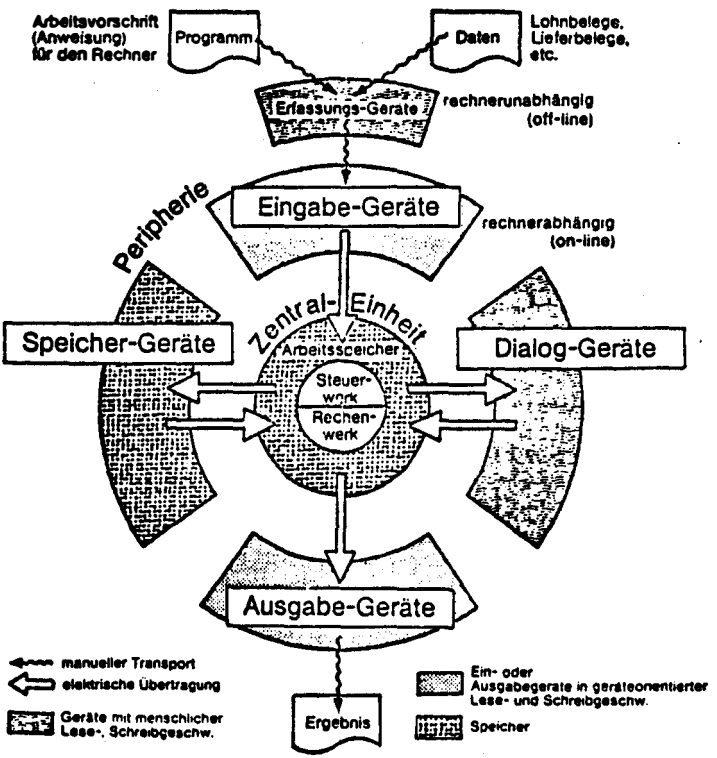
1 Kilobyte (= 1 Kb) = 1024 Bytes

Der ASCII-Code geht davon aus, daß jedes Byte, also jede Folge von 8 bits (von 00000000 bis 11111111 - dies sind 256 verschiedene Möglichkeiten) durch maximal zweistellige Hexadezimalzahlen darstellbar ist - von 00 bis FF. Nähere Einzelheiten dazu brauchen hier nicht besprochen zu werden. Eine ASCII - Code - Tabelle findet der interessierte Leser im Benutzerhandbuch des C64.

4.5 Hardware

Eine Rechenanlage ist im Prinzip immer so aufgebaut, wie es die folgende Abbildung schematisch zeigt :

Aufbau einer Rechenanlage



aus : WOLTERS...,a.a.O., Seite 5ø5

Die Informationen, die dem Rechner zugänglich gemacht werden sollen, stehen zunächst auf einem externen Datenträger (zum Beispiel auf Belegen,

Lochkarten u.ä.) und werden von dort durch sog. Eingabegeräte übernommen.

Über einen Zwischenspeicher gelangen sie dann in den Arbeitsspeicher der Zentraleinheit des Rechners und können von dort aus wiederum auf externen Speichern (sog. periphere Speicher) untergebracht werden.

Mit Dialoggeräten kann in einen laufenden Datenverarbeitungsprozeß eingegriffen werden; Ergebnisse des Prozesses werden dann auf Ausgabegeräten sichtbar.

Bei Kleinrechenanlagen ist das wichtigste Eingabegerät die Tastatur . Über die einzelnen Tasten (Symbol- und Steuertasten) des Commodore C64 soll hier nicht gesprochen werden; wir verweisen wieder auf das Benutzerhandbuch.

Die wichtigsten Ausgabegeräte sind bei einer Kleinrechenanlage der Bildschirm oder ein Drucker .

Von besonderer Bedeutung sind die externen Speicher der Peripherie. Informationen, die auf Dauer aufgehoben werden sollen - seien es Programme oder Datenbestände - müssen auf solche externen Speicher übetragen werden, weil sie beim Ausschalten des Rechners verloren gehen, wenn sie sich nur im Arbeitsspeicher befinden.

Die wichtigsten derartiger Speicher sind bei Kleinrechenanlagen Magnetbandkassetten (sie entsprechen den handelsüblichen Tonbandkassetten) und Disketten .

Die entsprechenden Speichergeräte sind das Kassettengerät bzw. das Diskettenlaufwerk.

Das Speichern auf eine Diskette ("Schreiben") bzw. das Laden von der Diskette ("Lesen") geht sehr viel schneller als dies bei einem Kassettenspeicher möglich ist, weshalb der fortgeschrittenere Computerbenutzer auf jeden Fall

die Diskette als externen Speicher vorzieht. Allerdings sind Diskettenlaufwerke deutlich teurer als Kassettengeräte.

Für Kassetten- wie für Diskettenspeicher gilt, daß gespeicherte Informationen überschrieben, d.h. durch andere ersetzt oder einfach gelöscht werden können.

Bleibt noch kurz über die Zentraleinheit des Rechners zu sprechen, die sog. CPU (= Central Processing Unit) : Hier werden die eingegebenen Informationen (Daten und Anweisungen) erfaßt und für die Dauer des Verarbeitungsprozesses gespeichert, hier findet dieser Prozeß der Datenverarbeitung selbst statt und hier wird auch die Ausgabe der Ergebnisse vorbereitet.

Diesen Aufgabenstellungen entsprechend wird die Zentraleinheit in die folgenden Bereiche unterteilt :

- Arbeitsspeicher
- Rechenwerk
- Steuerwerk

Im Arbeitsspeicher werden die zu verarbeitenden Daten und die zur Verarbeitung notwendigen Anweisungen (Programm) gespeichert, im Rechenwerk erfolgen die eigentlichen Verarbeitungsschritte und das Steuerwerk übernimmt die Organisation der einzelnen Arbeitsschritte. (Nähere Einzelheiten dazu in der einführenden Veröffentlichung von SACHT,H.-J. : 1981).

Kapitel 5 : BASIC-Grundlagen

5.1 Vorbemerkung

Die Demonstration dessen, was unter künstlicher Intelligenz verstanden werden kann, anhand geeigneter Computerprogramme für den Commodore C64, erfordert eine kurze Beschäftigung mit der Programmiersprache BASIC.

BASIC ist derzeit diejenige Programmiersprache, die bei Homecomputern die weiteste Verbreitung erfahren hat und genau dies ist der Grund, warum wir die folgenden Programme in dieser Sprache präsentieren. Allerdings ist mit Nachdruck darauf hinzuweisen, daß vor allem für komplexere Problemstellungen andere Programmiersprachen besser geeignet sind.

Da es aber im folgenden nicht darauf ankommt, optimal funktionierende Computerprogramme zu erstellen, sondern nur darauf, verständliche und nachvollziehbare Programme zu präsentieren, bedienen wir uns derjenigen Sprache, die am weitesten verbreitet ist und zudem den Vorteil aufweist, leicht verständlich (und leicht erlernbar) zu sein.

Natürlich soll in diesem Kapitel kein kompletter BASIC-Kurs angeboten werden, sondern wir müssen uns mit einem zusammenfassenden Überblick über diejenigen Sprachelemente begnügen, die im folgenden immer wieder benutzt werden.

Wir gehen dabei davon aus, daß der Leser schon über gewisse Grundkenntnisse verfügt, so daß dieses Kapitel quasi der "Auffrischung" dieser

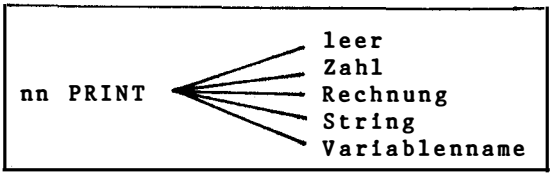
Kenntnisse dient. Derjenige Leser, der noch nie in BASIC programmiert hat, sollte an dieser Stelle die Lektüre dieses Buches unterbrechen und ein einführendes BASIC-Lehrbuch studieren. Derjenige hingegen, der schon perfekter BASIC-Programmierer ist, mag dieses Kapitel getrost überblättern.

5.2 Ergebnisausgabe

Jedes BASIC-Programm besteht aus einer Folge von Anweisungen (Statements), die dem Rechner zeilenweise einzugeben sind. Eine Programmzeile nennt man Satz .

Jeder Satz in BASIC benötigt eine Satznummer und ist mit der RETURN-Taste abzuschließen. In einem Satz können mehrere Statements stehen, die dann durch Doppelpunkte voneinander zu trennen sind.

Das wesentliche Statement zur Ergebnisausgabe ist das PRINT-Statement, das in allgemeiner Schreibweise folgendermaßen aussieht :



Die verschiedenen Möglichkeiten, die in der allgemeinen Statement-Darstellung genannt wurden (mit Ausnahme der Möglichkeit "leer") können in einem PRINT-Statement mehrfach und auch gemischt verwendet werden. Diese verschiedenen Ausdrücke sind dann durch Kommata oder durch Strichpunkte voneinander zu trennen.

Wird mit Kommata getrennt, so rücken die Ergebnisse auseinander; es erfolgt jeweils ein horizontaler Vorschub in die nächste Bildschirmzone ; wird hingegen der Strichpunkt als Trenner benutzt, geht es unmittelbar weiter.

Jedes PRINT-Statement bewirkt in der Regel einen Zeilenvorschub ; steht das Wort PRINT allein (Möglichkeit "leer"), so erfolgt nur ein Zeilenvorschub (Ausgabe einer Leerzeile). Ein Komma oder ein Strichpunkt am Ende eines PRINT-Statements unterdrückt den Zeilenvorschub.

Die Ergebnisausdrucke, die durch das PRINT-Statement erzeugt werden, beginnen jeweils am linken Bildschirmrand. Will man dies ändern, kann man sich der TAB-Funktion bedienen :

```
10 PRINT TAB(10) 3
```

bewirkt, daß die Zahl 3 zehn Stellen nach rechts verschoben ausgegeben wird.

Jedes BASIC-Programm wird mit dem folgenden Statement beendet :

```
nn END
```

Mit diesen beiden Statements zusammen (PRINT und END) kann man schon erste kleinere Programme entwerfen, die insbesondere zur Lösung von einfachen Rechnaufgaben verwendet werden können (im PRINT-Statement kann ja, wie oben angegeben wurde, auch gerechnet werden).

Programme werden vom Rechner erst dann abgearbeitet, wenn sie mit dem folgenden Kommando gestartet werden :

```
RUN
```

Es gilt dabei die folgende generelle Regel :

Kommandos erhalten keine Satznummer !

Will man ein Programm wieder auf dem Bildschirm erscheinen lassen, zum Beispiel weil es über den oberen Bildschirmrand hinausgewandert ist, so benötigt man das folgende Kommando :

```
LIST
```

Häufig ist es sinnvoll, in Programme Kommentare, Erläuterungen, Zwischenüberschriften einzufügen oder es mit einer Überschrift zu versehen. Dies ist mit dem folgenden Statement möglich :

```
nn REM Text
```

Dieses REM-Statement taucht mit seinem Text beim Listen eines Programms immer wieder auf, hat aber auf die Programmabarbeitung selbst keinen Einfluß.

Wenn ein Programm im Arbeitsspeicher gelöscht werden soll, so benötigt man das folgende Kommando

```
NEW
```

Das jeweilige Programm steht nach Eingabe dieses Kommandos nicht mehr zur Verfügung (also ist Vorsicht geboten!).

5.3 Wertzuweisungen

Wir haben schon darauf hingewiesen, daß die Speicherplätze im Rechner mit Namen zu versehen sind. In diesen Speicherplätzen (Feldern) können dann die Werte von Variablen gespeichert werden. Bei der Namensvergabe muß auf den Typ der jeweiligen Variablen geachtet werden.

Sinnvollerweise unterscheidet man dabei zwischen numerischen Variablen und Stringvariablen , wobei erstere wiederum in reelle und ganzzahlige Variablen unterschieden werden können. Reelle Variablen haben als Werte Dezimalzahlen (in BASIC verwendet man immer einen Punkt zum Abtrennen von Dezimalstellen), ganzzahlige Variablen hingegen nur ganze Zahlen.

Der Name einer reellen Variablen besteht aus einem oder zwei alphanumerischen Zeichen, wobei das erste immer ein Buchstabe sein muß. Die Namen ganzzahliger Variablen werden zusätzlich mit dem %-Zeichen, die Namen von Stringvariablen mit dem $-Zeichen versehen.

Nach diesen Erläuterungen können wir das LET-Statement besprechen, das der Wertzuweisung dient :

```
nn LET Variablenname =  <  Zahl
                           Rechnung
                           String
                           Variablenname
```

Das Wort LET darf auch entfallen.

Aus der zweiten Möglichkeit geht hervor, daß mit diesem Statement, wie übrigens auch schon mit dem PRINT-Statement, gerechnet werden kann. Dabei werden die folgenden arithmetischen Operatoren

benutzt :

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
↑	Potenzierung

Die Rangfolge der Operatoren entspricht der aus der Schulmathematik bekannten; bei Bedarf kann durch geeignete Klammernsetzung davon abgewichen werden.

Viele Berechnungen können durch eingebaute Funktionen erleichtert werden. Die allgemeine Schreibweise eines solchen Funktionsaufrufs lautet folgendermaßen :

Variablenname = Funktionsname(Argument(e))

Die wichtigsten dieser Funktionen sind die folgenden :

SQR(Zahl)	:	berechnet die Quadratwurzel aus Zahl
INT(Zahl)	:	bestimmt den größten, in Zahl enthaltenen ganzzahligen Wert
RND(1)	:	bestimmt eine Zufallszahl zwischen 0 bis unter 1
CHR$(Zahl)	:	bestimmt das ASCII-Codezeichen der Codezahl Zahl

Zur zuletztgenannten CHR$-Funktion soll ein wichtiger Anwendungsfall genannt werden :

Die Anweisung

```
nn PRINT CHR$(147)
```

kann beim Commodore C64 dazu benutzt werden, im laufenden Programm den Bildschirm zu löschen. Vor jeder Ergebnisausgabe ist dies sehr nützlich.

5.4 Informationseingabe

Konkrete Datenverarbeitungsprobleme erfordern es, daß Daten eingegeben werden. In BASIC eignet sich dazu ganz hervorragend das folgende Statement :

```
nn INPUT ("Text";) Variablenliste
```

Gelangt der Rechner bei der Programmabarbeitung an dieses Statement, so unterbricht er sie und produziert auf dem Bildschirm ein Fragezeichen (vorher den eventuell im Statement vorhandenen Text, der aber auch weggelassen werden kann). Er erwartet dann vom Benutzer so viele Werte (durch Kommata zu trennen), wie Variablennamen in der Variablenliste genannt sind.

Ähnlich wie das INPUT-Statement erwartet auch das folgende Statement eine Informationseingabe, die aber nur aus der Betätigung einer einzigen Taste besteht und nicht durch die RETURN-Taste abgeschlossen werden muß.

```
nn GET Variablenname$
```

Zur Eingabe größerer Informationsmengen ist das INPUT-Statement nicht so gut geeignet. Besser geeignet ist das DATA-Statement in Verbindung mit dem READ-Statement :

```
nn DATA Wert1,Wert2,Wert3, ...
```

```
nn READ Name1,Name2,Name3, ...
```

Die Lese-Anweisung READ bewirkt, daß der erste Wert des DATA-Statements in das Feld Name1, der zweite Wert in das Feld Name2, der dritte Wert in das Feld Name3 gelesen wird usw.

Es dürfen in READ-Statements nicht mehr Namen auftauchen als in DATA-Statements Werte vorhanden sind, wohl aber ist es möglich, weniger Werte als vorhanden zu lesen. Eventuell folgende READ - Statements fahren in der (oder den) DATA - Liste(n) dort fort zu lesen, wo die vorhergehende READ-Anweisung aufgehört hat.

Bei großen Datenbeständen empfiehlt es sich, unter einem Variablennamen jeweils mehrere Werte zu erfassen und nicht wie bisher immer nur einen Wert. Dies gelingt durch Verwendung <u>indizierter Variablen</u> .

In der Programmiersprache BASIC sieht das z.B. folgendermaßen aus :

X(0), X(1), X(2), ... allgemein z.B. X(I)

Bei Verwendung solcher indizierten Variablen muß allerdings dem Rechner vor deren erster Verwendung mitgeteilt werden, wieviele Speicherplätze für die jeweilige(n) Variable(n) freigehalten werden sollen. Dies leistet das folgende Statement :

```
nn DIM Name1(Zahl1),Name2(Zahl2), ...
```

Es kann auch doppelt indiziert werden, also zum

Beispiel :

```
10 DIM X(2,3)
```

In diesem Beispiel werden für die Variable X 3*4 = 12 Speicherplätze freigehalten, nämlich die Plätze

X(0,0)	X(0,1)	X(0,2)	X(0,3)
X(1,0)	X(1,1)	X(1,2)	X(1,3)
X(2,0)	X(2,1)	X(2,2)	X(2,3)

5.5 Programmverzweigungen

Solange in einem BASIC-Programm noch keine Verzweigungen auftreten (sog. Sprünge), erfolgt die Abarbeitung in der Reihenfolge der angegebenen Satznummern. Will man von dieser Reihenfolge abweichen, so sind Programmsprünge erforderlich. Wir unterscheiden unbedingte Sprünge von bedingten Sprüngen .

Der unbedingte Sprung wird mit dem folgenden Statement möglich :

```
nn GOTO mm
```

Erreicht die Programmsteuerung dieses Statement, so erfolgt ein Sprung zum Satz mit der Satznummer mm.

Die bedingte Programmverzweigung wird mit dem folgenden Statement ermöglicht :

```
nn IF logische Bedingung THEN Anweisung
```

Hier wird eine Abfrage veranlaßt : Wird die logische Bedingung, die dem Befehlswort IF folgt, erfüllt, so wird die hinter dem THEN stehende Anweisung ausgeführt; ist dies hingegen nicht der Fall, so geht die Programmsteuerung zu dem Satz über, der dem IF-Statement folgt.

Hinter dem Schlüsselwort THEN können noch weitere Anweisungen im gleichen Satz stehen; auch diese werden nur dann ausgeführt, wenn die Bedingung erfüllt ist.

Im IF ... THEN-Statement können sehr gut auch mit Hilfe von String-Variablen logische Bedingungen aufgebaut werden. Dies verdeutlicht das folgende Beispiel :

```
10 B$ = "SEHR GUT"
20 PRINT B$
30 PRINT
40 INPUT "NOCHMAL (JA ODER NEIN)";A$
50 IF A$ = "JA" THEN GOTO 20
60 PRINT
70 PRINT "ENDE DER AUSGABE":END
```

Häufig ist es sinnvoll, daß bestimmte Programmsegmente ganz aus dem eigentlichen Hauptprogramm ausgelagert werden, um dann von diesem aus mehrfach (bei Bedarf) angesprungen zu werden.

Solche ausgelagerten Programmteile nennt man Unterprogramme . Der Sprung vom Hauptprogramm ins Unterprogramm erfordert das folgende Statement :

```
nn GOSUB mm
```

Dieses Statement bewirkt einen Sprung in das Unterprogramm, das mit der Satznummer mm beginnt. Ist dann dieses Unterprogramm abgearbeitet, muß ein Rücksprung in das Hauptprogramm erfolgen. Zu diesem Zweck wird das Unterprogramm mit dem folgenden Statement beschlossen :

```
nn RETURN
```

Dieses Statement bewirkt, daß ein Rücksprung an diejenige Stelle des Hauptprogramms erfolgt, die direkt hinter der entsprechenden GOSUB-Anweisung folgt.

Besonders interessante Möglichkeiten eröffnet das folgende Statement :

```
nn ON Variablenname GOTO m1,m2,m3,...
```

Wird dieses Statement erreicht, so erfolgt ein Sprung zum Satz m1, wenn Variablenname den Wert 1 hat, ein Sprung zum Satz m2, wenn Variablenname den Wert 2 hat, usw. Mit diesem Statement sind also Mehrfachverzweigungen möglich.

Wir haben weiter oben gesehen, daß durch geeignete Verwendung des IF ... THEN-Statements erreicht werden kann, daß bestimmte Programmteile mehrfach durchlaufen werden können. Man spricht von sog. Schleifen . Derartige Schleifen können mit den folgenden Statements sehr einfach erzeugt werden :

```
nn FOR Name = Anfang TO Ende (STEP Schrittweite)
```

und

```
nn NEXT Name
```

An der Stelle Name steht dabei der Name eine Variablen, die als Laufvariable bezeichnet wird. Sie taucht auch wieder im NEXT-Statement auf und muß vom reellen Typ sein. Anfang, Ende und Schrittweite können Zahlen, Variablen oder arithmetische Ausdrücke sein.

Gelangt der Rechner an das FOR-Statement, so wird zunächst die Laufvariable mit dem Anfangswert belegt und der Programmteil zwischen dem FOR ... TO- und dem NEXT-Statement wird mit diesem Wert der Laufvariablen durchlaufen.

Bei Erreichen des NEXT-Statements wird die Laufvariable mit dem nächsten Wert, nämlich mit aktuellem Wert + Schrittweite belegt und wieder der selbe Programmteil abgearbeitet, jetzt mit diesem veränderten Wert.

Dies erfolgt so lange, bis bei Erreichen des NEXT-Statements durch weitere Addition der Schrittweite der Endwert überschritten wird. Dann wird die Schleife verlassen und der nach dem NEXT-Statement folgende Satz bearbeitet.

STEP Schrittweite kann im FOR ... TO-Statement entfallen, wenn die Schrittweite gleich eins sein soll.

5.6 Stringbearbeitung

Wir haben schon erwähnt, daß für bestimmte Zwecke Funktionen zur Verfügung stehen. Eine Reihe solcher Funktionen können speziell zur Stringverarbeitung und -bearbeitung eingesetzt werden (String = Zeichenkette). Die wichtigsten dieser Funktionen sollen kurz vorgestellt werden.

CHR$(Zahl)

Diese Funktion bestimmt das ASCII-Codezeichen, das zur Codezahl Zahl gehört.

LEFT$(String,Zahl)

Diese Funktion schneidet aus String von links beginnend einen Teilstring ab, der so viele Symbole umfaßt, wie durch Zahl angegeben wird.

RIGHT$(String,Zahl)

Entsprechend schneidet diese Funktion einen Teilstring von rechts beginnend ab.

MID$(String,Zahl1,Zahl2)

Diese Funktion schneidet aus String beim Symbol Zahl1 beginnend einen Teilstring heraus, der so viele Symbole umfaßt, wie durch Zahl2 angegeben wird.

Für viele Anwendungsbereiche ist auch die folgende Funktion wichtig :

LEN(String)

Diese Funktion bestimmt die Anzahl der Symbole, die in String enthalten sind.

Die letzte Funktion, die hier betrachtet werden

soll, ist die folgende :

STR$(Zahl)

Diese Funktion wandelt Zahl in eine Zeichenkette um.

5.7 Graphik

Bei den Programmen der folgenden Kapitel wird - wenn auch in bescheidenem Maße - von den Graphikmöglichkeiten des Commodore C64 Gebrauch gemacht. Hierzu bieten sich zunächst die Graphiksymbole an, die die C64-Tastatur aufweist.

Beispielsweise findet sich auf der Vorderseite der Q-Taste das Zeichen ●. Dieses Zeichen kann mit dem folgenden Programm "gezeichnet" werden :

```
10 PRINT " [SHIFT]    und    [Q] "
20 END
```

Der Satz 10 erscheint auf dem Bildschirm folgendermaßen :

```
10 PRINT "●"
```

Entsprechend können die anderen Graphiksymbole der Tastatur verwendet werden. In Verbindung mit der TAB-Funktion und unter Verwendung von Programmschleifen oder von Unterprogrammen können auf diese Weise Striche, Balken, Umrandungen u.ä. erzeugt werden.

Häufig ist es sinnvoll, wenn bei dieser Art der Graphikprogrammierung bestimmte Stellen (Positionen) des Bildschirms gezielt angesprochen

werden können. Dies ist sehr leicht folgendermaßen möglich :

Man muß sich den Bildschirm aufgeteilt denken in 40 Spalten (von links nach rechts 0 bis 39) und 25 Zeilen (von oben nach unten 0 bis 24). Somit ergeben sich in dieser sog. Normalgraphik 1000 Bildschirmpositionen (diese Normalgraphik ist von der sog. hochauflösenden Graphik zu unterscheiden).

Diese 1000 Positionen werden zeilenweise durchnumeriert von der Adressennummer 1024 links oben bis zur Adresse 2023 rechts unten. Jede dieser Adressen, d.h. jede einzelnen Bildschirmposition kann nun mit einem der Graphikzeichen oder mit irgendeinem der anderen Symbole belegt werden. Dazu benötigt man die folgende Anweisung :

```
nn POKE Adresse,Zahl
```

Unter Adresse steht eine Zahl zwischen 1024 und 2023 (s.o.), unter Zahl steht die ASCII-Codezahl desjenigen Zeichens, das in der entsprechenden Bildschirmposition erscheinen soll.

Die Adresse einer beliebigen Bildschirmposition findet man mit der folgenden Formel :

Adresse = 1024 + S + 40*Z

Dabei ist S die Spalte und Z die Zeile, in der das Symbol erscheinen soll.

Will man also beispielsweise in der Bildschirmposition, die durch Spalte 20 und Zeile 10 gekennzeichnet wird, ein Kügelchen zeichnen (Codezahl 81), so erhalten wir die Adresse wie folgt :

Adresse = 1024 + 20 + 40*10 = 1444

und das Zeichenprogramm lautet dann :

```
10 POKE 1444,81
20 END
```

Zusätzlich kann man bei solchen "Zeichnungen" auch die Farbe der Ausgabe beeinflussen :

Den Bildschirmadressen (1024 bis 2023) sind Farbadressen zugeordnet, die die Adressen 55296 bis 56295 aufweisen. Hat man beispielsweise die Bildschirmadresse 1444 ausgewählt, so ergibt sich die zugehörige Farbadresse , indem man rechnet :

Farbadresse = Bildschirmadresse + 54272

also :

Farbadresse = 1444 + 54272 = 55716

Die "Einfärbung" einer bestimmten Bildschirmadresse erfolgt dann mit der folgenden Anweisung :

```
nn POKE Farbadresse,Zahl
```

An der Stelle Zahl steht ein Wert zwischen 0 und 15 mit folgender Bedeutung :

Zahl	Farbe	Zahl	Farbe
0	schwarz	8	orange
1	weiß	9	braun
2	rot	10	hellrot
3	türkis	11	grau 1
4	violett	12	grau 2
5	grün	13	hellgrün
6	blau	14	hellblau
7	gelb	15	grau 3

Abschließend sei darauf hingewiesen, daß auch innerhalb von PRINT-Anweisungen ein Farbwechsel bei der Ausgabe von Informationen erzielt werden kann. Dies gelingt unter Benutzung der Zifferntasten zusammen mit der CTRL-Taste bzw. mit der C= -Taste.

Dabei bedeuten CTRL und 1 bis CTRL und 8 das gleiche wie die Zahlen 0 bis 7 in der obigen Farbtabelle, bzw. C= und 1 bis C= und 8 das gleiche wie die Zahlen 8 bis 15 in der obigen Tabelle.

DRITTER TEIL:

Programme für den C 64

Kapitel 6 : Einführung

6.1 Zielsetzungen

Künstliche Intelligenz ist ein kompliziertes Gebiet. Entsprechend kompliziert fallen "intelligente" Programme aus. Wenn ein Computer sich tatsächlich so verhalten soll, als sei er intelligent, so benötigt man in der Regel sehr umfangreiche Programme, die zum Teil dann auch sehr komplexe Strukturen aufweisen.

Hier geht es deshalb darum, die wichtigsten Funktionsprinzipien derartiger Programme an überschaubaren, d.h. möglichst kurzen Beispielen vorzustellen. Wir werden dabei nur die einfachen BASIC-Elemente verwenden, die im vorangegangenen Kapitel kurz besprochen wurden und weder auf Spezialitäten dieser Sprache zurückgreifen, noch uns mit anderen Programmiersprachen befassen.

Kompliziertere Programme könnte man nämlich recht gut in einer maschinennahen Sprache, zum Beispiel in ASSEMBLER programmieren, oder man könnte zumindest auf eine der traditionellen Sprachen wie FORTRAN oder COBOL zurückgreifen - wenn auch nicht bei Verwendung eines C64 - oder auf die speziell für Programme der künstlichen Intelligenz entwickelte Sprache LISP.

Aber auch mit einfachen BASIC-Programmen können sehr wohl die Grundprinzipien illustriert werden, um die es hier geht.

Allerdings sind auch diese Demonstrationsprogramme zum Teil schon so komplex, daß eine sorgfältige Problemanalyse zur Vorbereitung der Erstellung der jeweiligen Programme unentbehrlich ist. Eine derartige Problemanalyse ist nichts anderes als eine Vorstrukturierung der jeweiligen

Problemlösungswege, die notwendig wird, um zu überschaubaren, d.h. zu gut strukturierten Programmen zu gelangen.

Bei derartigen Vorstrukturierungen bedient man sich in der Regel sog. Flußdiagramme (auch Programmablaufpläne genannt), um sich den (oder die) Lösungsweg(e) graphisch zu veranschaulichen.

Mit dieser Vorstrukturierung wird deutlich, wie wichtig es ist, vor der Programmierung eines bestimmten Problems den Lösungsweg klar und in kleinstmögliche Schritte zerlegt vorzuzeichnen. Ganz generell gilt nämlich, daß nur dann ein brauchbares Programm entwickelt werden kann, wenn der Ersteller des Programms den Lösungsweg (korrekt) schon im Kopf hat.

Nur derjenige, der genau weiß, wie der Satz des Pythagoras lautet, der kann auch problemlos ein BASIC-Programm zur Berechnung der Grundseite in rechtwinkligen Dreiecken erstellen.

Wie geht man bei einer Problemanalyse vor ?

Wir wollen dies an drei Beispielen im folgenden Abschnitt demonstrieren.

6.2 Problemanalyse

Die Problemanalyse bei einer konkreten Aufgabenstellung orientiert sich an den folgenden Arbeitsschritten :

1. Präzisierung der Aufgabenstellung :

In einem ersten Schritt muß genau festgelegt werden, was eigentlich der Gegenstand der jeweiligen Fragestellung ist. Wenn beispielsweise gefordert wird, aus gegebenen statistischen Angaben zu den Monatseinkommen der Haushalte in der Bundesrepublik Deutschland das mittlere Einkommen auszurechnen, dann muß zunächst festgelegt werden, welcher Mittelwert berechnet werden soll : Ist das arithmetische Mittel gefragt, das geometrische Mittel, der Zentralwert, der Modus oder sonst ein Mittelwert ?

Entsprechende Fragen beziehen sich auf den Begriff "Haushalt", auf den Begriff "Monatseinkommen" etc., d.h. auch diesbezüglich sind Präzisierungen erforderlich.

2. Eignet sich der Rechner zur Problemlösung ?

Dem ein oder anderen Leser mag diese Frage seltsam vorkommen, hat er doch gehört, daß Computer so gut wie alle Probleme lösen können, wenn sie korrekt programmiert sind. Gleichwohl ist es nicht überflüssig, diese Frage zu stellen. Es sind Probleme vorstellbar, die beispielsweise nur auf intuitivem Wege bearbeitet werden können - wenn überhaupt. Der Computereinsatz wäre dann weniger empfehlenswert.

3. Grobstruktur des Problems

Die Problemanalyse wird durch eine vorläufige Grobstrukturierung des Problems vorbereitet.

Gemeint ist damit, daß zunächst die folgenden drei Fragen beantwortet werden :

a) Welche Eingabe-Informationen benötigt der Rechner ?

b) Welche Verarbeitungsschritte folgen ?

c) Welche Ausgaben sollen erfolgen ?

Diese drei Fragen faßt man im sog.

E V A - Prinzip

zusammen. Es besagt, daß die Problemanalyse die nachstehende Abfolge von Schritten diskutiert :

E	Eingabe
V	Verarbeitung
A	Ausgabe

Diese Arbeitsschritte sollen zusammen mit den übrigen Aufgaben der Problemanalyse im folgenden an Beispielen illustriert werden.

<u>Beispiel 1</u>

Berechnung eines arithmetischen Mittels aus beliebigen, einzugebenden Zahlenwerten :

Das arithmetische Mittel ist definiert als die Summe aller Werte dividiert durch deren Anzahl, also :

$$AM = \frac{\sum x_i}{N}$$

Wenn ein Programm für beliebige Werte ein derartiges arithmetisches Mittel berechnen soll, dann ist gemäß des EVA-Prinzips zunächst darauf zu achten, daß eine beliebige Zahl von Werten eingegeben werden kann.

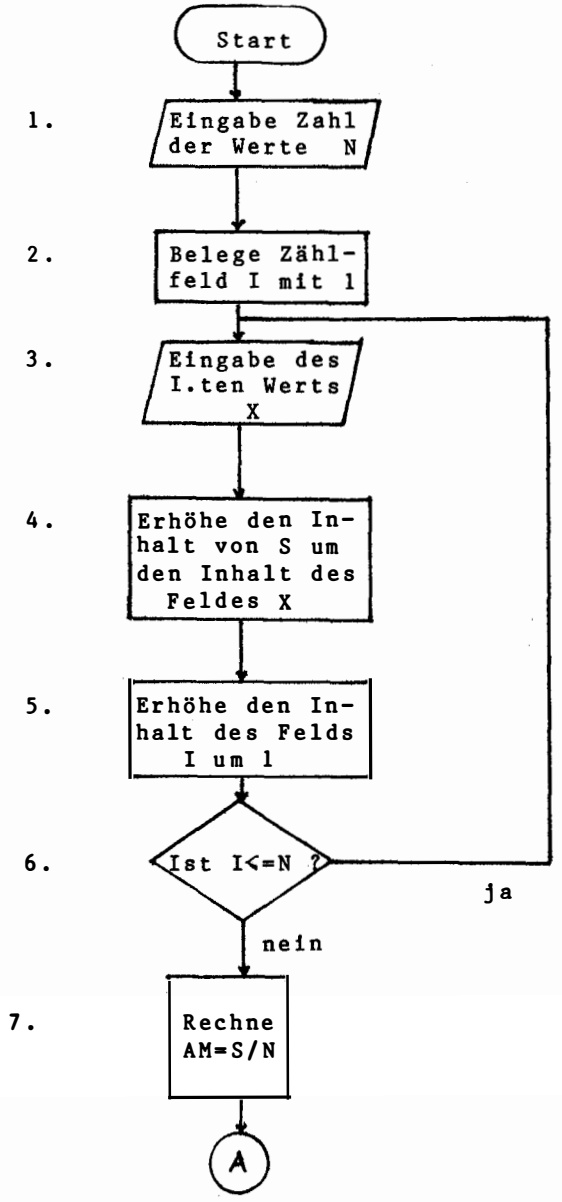
Dies erreichen wir dadurch, daß das Programm, das entwickelt werden soll, als erstes den Programmbenutzer fragt, wieviele Werte er einzugeben wünscht, wieviele also in die Mittelwertberechnung eingehen sollen. Ist dies bekannt, kann dann eine Schleife programmiert werden, die genau so viele Werte anfordert, wie eingegeben werden sollen.

Diese Werte müssen - entsprechend der zuständigen Berechnungsformel - aufaddiert werden. Ist der Eingabe- und Additionsprozeß beendet, so wird die erreichte Summe durch die Zahl der Werte dividiert (damit ist das V im EVA-Prinzip angesprochen).

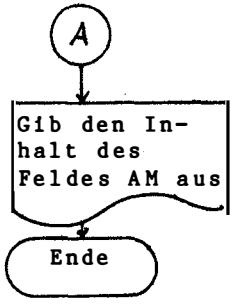
Der Ausgabeschritt (A im EVA-Prinzip) besteht dann einfach darin, das errechnete Ergebnis auf dem Bildschirm auszugeben.

Damit gelangen wir zu dem folgenden Ablaufplan :

Ablaufplan Mittelwertberechnung :



8.



Dieses Flußdiagramm zeigt in anschaulicher Weise, was während des Ablaufs des zu entwickelnden Programms zu geschehen hat. Nähere Erläuterungen dazu sind sicherlich entbehrlich.

Zur Prüfung eines derartigen Programmablaufplans, und damit zur Prognose, ob das spätere Programm das tun wird, was es auch tun soll, ist eine Prüfung der Felderbesetzungen während des Programmablaufs anhand von Probedaten sinnvoll.

Im Programm werden laut Flußdiagramm die folgenden Speicherstellen benutzt :

N, I, X, S, AM

Die Belegungen dieser Felder und die Veränderungen dieser Belegungen lassen sich anhand des folgenden Beispiels darstellen :

Nehmen wir einmal an, wir wollten den Mittelwert aus den folgenden vier Zahlen per Programm ausrechnen :

11, 17, 9, 13

Felderbelegungen :

N	I	X	S	AM	Position im Flußdiagramm
4					1.
	1				2.
		11			3.
			11		4.
	2				5.
2 ist kleiner gleich 4					6.
		17			3.
			28		4.
	3				5.
3 ist kleiner gleich 4					6.
		9			3.
			37		4.
	4				5.
4 ist kleiner gleich 4					6.
		13			3.
			50		4.
	5				5.
5 ist nicht kleiner gleich 4					6.
				12.25	7.

Diese ausführliche Beschreibung der Felderbelegungen verdeutlicht, daß in dem Feld, dessen Inhalt im 8. Schritt ausgegeben werden soll, nämlich im Feld AM tatsächlich der korrekte Wert 12.25 steht, das Ergebnis des Probelaufs.

Das entsprechende BASIC-Programm sieht folgendermaßen aus :

```
10 PRINT CHR$(147)
20 INPUT "WIEVIELE WERTE : ";N
30 FOR I = 1 TO N
40 PRINT I;".WERT : ";:INPUT X
50 S = S+X
60 NEXT I
70 PRINT:PRINT:PRINT
80 AM = S/N
90 PRINT "MITTELWERT = ";AM
100 END
```

Nach dieser ausführlichen Darstellung fassen wir uns bei den folgenden Beispielen sicherlich kürzer.

Beispiel 2

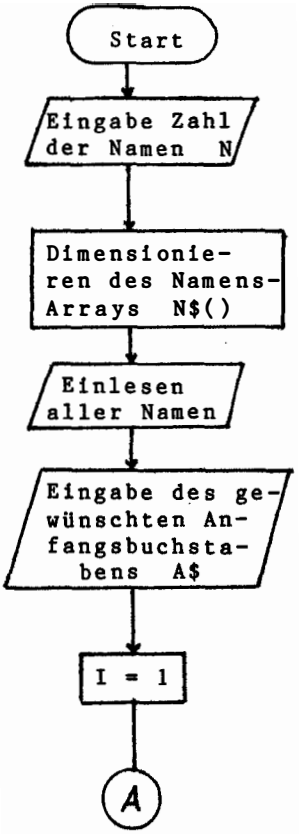
Aus einem Vorrat an Namen, die beispielsweise über DATA-Statements eingegeben werden, sollen alle diejenigen herausgesucht werden, die mit einem bestimmten, frei wählbaren Anfangsbuchstaben beginnen.

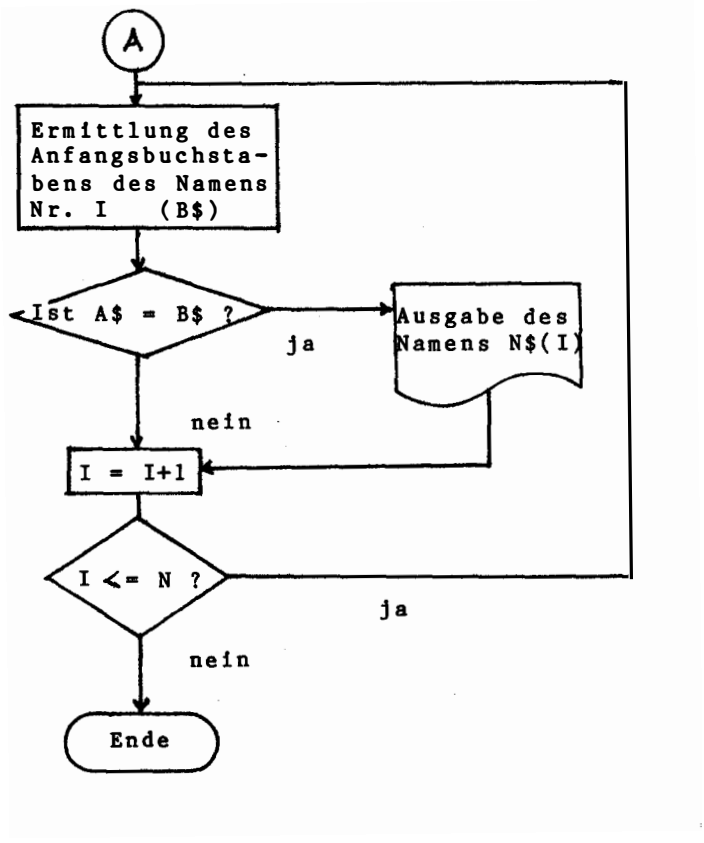
Der Eingabeschritt besteht darin, die Namen und den gewünschten Anfangsbuchstaben einzugeben. Es wird dabei in diesem Beispiel vorausgesetzt, die Anzahl der Namen sei bekannt (zum Beispiel N = 15).

Im Verarbeitungsschritt ist von jedem Namen der Anfangsbuchstabe abzutrennen. Dieser Teilstring wird dann mit dem gewünschten Buchstaben verglichen. Wird bei diesem Vergleich keine Übereinstimmung festgestellt, so ist der nächste Name zu prüfen.

Der Ausgabeschritt besteht darin, daß dann, wenn Übereinstimmung festgestellt wurde, der betreffende Name ausgegeben wird. Auch danach wird dann der nächste Name geprüft.

Flußdiagramm zur Auswahl von Namen gemäß des Anfangsbuchstabens :




A
Ermittlung des Anfangsbuchstabens des Namens Nr. I (B$)
Ist A$ = B$?
ja
Ausgabe des Namens N$(I)
nein
I = I+1
I <= N ?
ja
nein
Ende

Das entsprechende BASIC-Programm könnte folgendermaßen aussehen :

```
10 PRINT CHR$(147)
20 N = 15
30 DIM N$(N)
40 FOR I=1 TO N:READ N$(I):NEXT I
50 INPUT "ANFANGSBUCHSTABE : ";A$
60 PRINT:PRINT:PRINT
70 PRINT "DIE FOLGENDEN NAMEN HABEN"
80 PRINT "DEN ANFANGSBUCHSTABEN ";A$
90 PRINT:PRINT:PRINT
100 FOR I=1 TO N
110 B$ = LEFT$(N$(I),1)
120 IF A$ = B$ THEN PRINT N$(I)
130 NEXT I
140 PRINT:PRINT:PRINT
150 PRINT "ENDE DER AUSGABE"
500 DATA HANS,FRITZ,EMIL,WERNER,OTTO
510 DATA SUSI,BERTA,EVA,OLGA,KARIN
520 DATA UWE,FRANZ,HEIKE,SABINE,LOTTE
999 END
```

Auch bei dieser einfachen Problemstellung dürften nähere Erläuterungen entbehrlich sein. Deshalb haben wir auf die Ablaufkontrolle durch die Verfolgung der Felderbesetzungen verzichtet.

Beispiel 3

In einem letzten Beispiel wollen wir ein in der Datenverarbeitung sehr wichtiges Problem vorstellen : Es geht darum, einen beliebigen Datenbestand der Größe nach zu sortieren.

" Sortieren " bedeutet, daß je zwei Werte aus einem Datenbestand miteinander verglichen werden müssen. Man vergleicht in einer ersten Vergleichsrunde den ersten Wert Schritt für

Schritt mit allen übrigen. Ist dabei der erste Wert größer als einer der Vergleichswerte, so müssen beide miteinander vertauscht werden (es wird dabei untertellt, daß am Ende des Sortierprozesses der kleinste Wert am Anfang, der größte am Ende der sortierten Reihe stehen soll).

Mit dieser ersten Vergleichsrunde wird erreicht, daß am Ende dieser Runde an erster Stelle der insgesamt kleinste Wert steht.

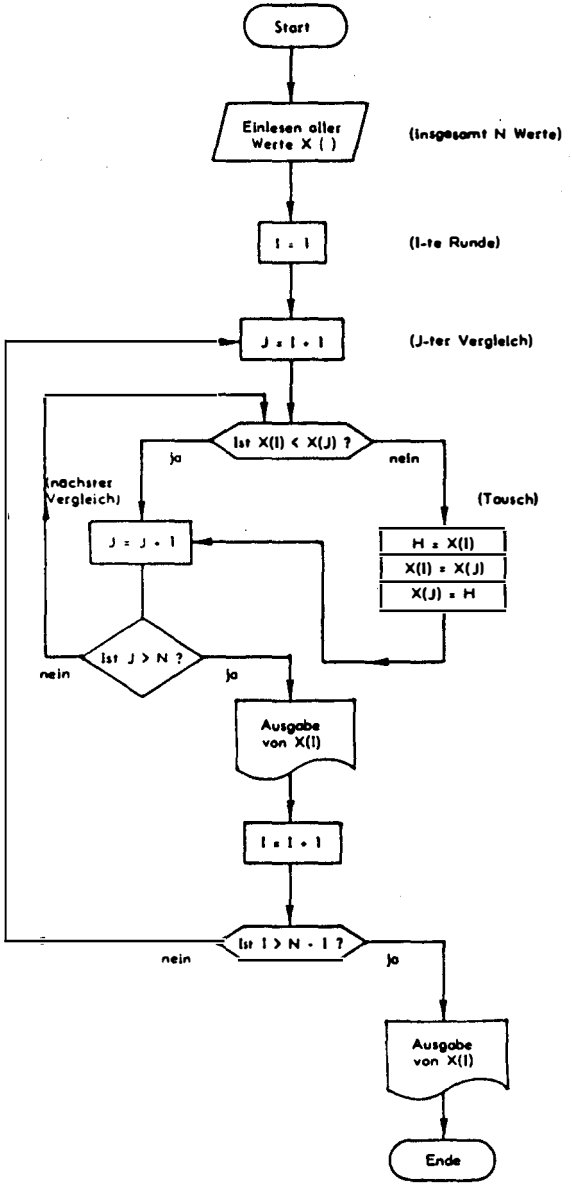
In einer zweiten Runde vergleicht man den jetzt an zweiter Stelle stehenden Wert mit allen übrigen, außer mit dem ersten Wert. Gegebenenfalls wird wieder getauscht, so daß am Ende der zweiten Runde an zweiter Stelle der insgesamt zweitkleinste Wert steht.

In der dritten Runde vergleicht man den an dritter Stelle stehenden Wert mit allen übrigen außer mit den ersten beiden. Am Ende dieser dritten Runde hat man dann den drittkleinsten Wert gefunden.

Auf diese dritte Runde folgen so lange weitere Vergleichsrunden wie überhaupt Vergleiche möglich sind : Haben wir beispielsweise fünf Werte zu sortieren, so sind vier derartige Vergleichsrunden möglich. In der ersten Runde gibt es vier, in der zweien drei, in der dritten zwei und in der vierten Runde einen Vergleich(e).

Das Flußdiagramm zu dieser Problemlösung sieht folgendermaßen aus :

Flußdiagramm:



Das BASIC-Programm, das keiner näheren Erläuterung bedarf, sieht folgendermaßen aus :

```
10 PRINT CHR$(147)
20 INPUT "WIEVIELE DATEN : ";N
30 DIM X(N)
40 REM ***** EINGABE DER DATEN *****
50 FOR I=1 TO N
60 PRINT I;".WERT : ";:INPUT X(I)
70 NEXT I
80 PRINT:PRINT:PRINT
90 REM ***** SORTIEREN *************
100 FOR I = 1 TO N-1
110 FOR J = I+1 TO N
120 IF X(I) ²= X(J) THEN GOTO 150
130 REM ***** TAUSCH ***************
140 H=X(I):X(I)=X(J):X(J)=H
150 NEXT J
160 REM ***** AUSGABE **************
170 PRINT X(I)
180 NEXT I
190 REM ***** LETZTER WERT *********
200 PRINT X(N)
210 PRINT:PRINT:PRINT
220 PRINT "ENDE DER AUSGABE":END
```

Die Analyse dieses Programms bereitet dem Leser sicherlich keine unüberwindlichen Schwierigkeiten. Allenfalls ist ein Wort zum Satz 140 angebracht, in dem der Austausch von zwei Feldinhalten vollzogen wird, wenn dies erforderlich ist :

Ein solcher Austausch macht die Einrichtung eines Hilfsfeldes erforderlich (wir haben es H genannt), in dem der Inhalt des Feldes X(I) "aufbewahrt" wird, während jenes den Inhalt von X(J) aufnimmt. Diese Aufbewahrung ist erforderlich, damit das Feld X(J) dann diesen Inhalt von H übernehmen kann, und somit den Inhalt des Feldes X(I) erhält. Dann hat sich der Tausch korrekt vollzogen.

6.3 Die Demonstrationsbeispiele

Entsprechend der Auflistung der Hauptgebiete der künstlichen Intelligenz, wie sie sich im ersten Teil findet, werden im folgenden Demonstrationsbeispiele zu den folgenden Einsatzbereichen vorgestellt, die wir jeweils mit einem Stichwort bezeichnen :

1. Expertensystem
2. Auskunftssystem
3. Suchprogramm
4. Entscheidungsprogramm
5. Erkennen
6. Spielprogramm
7. Selbstlernendes Programm
8. Kunst
9. Dialog
10. Reaktion
11. Unterrichtsprogramm

Der Leser erkennt, daß ein wichtiger Bereich hier nicht genannt wurde, nämlich der Bereich der sog. Robotik . Dieser Bereich, der mit der programmgesteuerten Bewegung von Maschinen zu tun hat, ist deshalb nicht aufgenommen worden, weil er sich derzeit zu einem sehr wichtigen und umfangreichen, eigenen Gebiet entwickelt.

Zu den Fragen der Robotik wird in zunehmendem Maße spezielle Literatur veröffentlicht, so daß weitere Ausführungen hier entbehrlich sind. Zudem gilt, daß Programme der Robotik auf einem Kleinrechner wie dem Commodore C 64 nicht sinnvoll adaptierbar sind, weil man für derartige Programme größere Rechenanlagen benötigt.

Die folgenden Kapitel sind jeweils einem Illustrationsprogramm zu den oben genannten

Bereichen gewidmet. Jedes dieser Kapitel ist folgendermaßen aufgebaut :

1. Beschreibung der Aufgabenstellung :

In diesem Abschnitt wird skizzenartig dargestellt, worum es in dem jeweiligen Anwendungsbereich eigentlich geht, und was von Computern, wenn sie in diesem Bereich eingesetzt werden, erwartet werden kann.

2. Lösungsansätze :

In diesem Abschnitt wird unter eher theoretischen Gesichtspunkten die Frage erörtert, wie Programme aussehen müßten, die die in Abschnitt 1 genannten Aufgabenstellungen bewältigen können.

3. Das Beispiel :

In diesem dritten Abschnitt stellen wir das Beispiel vor, das wir ausgewählt haben, um den jeweiligen Anwendungsbereich zu illustrieren. Es sei nochmals darauf hingewiesen, daß es sich um einfache Illustrationsbeispiele handelt.

4. Problemanalyse :

Im diesem Abschnitt stellen wir die Überlegungen vor, die angestellt werden müssen, bevor man zu dem im dritten Abschnitt genannten Beispiel ein BASIC-Programm schreiben kann. Wir orientieren uns dabei an den Ausführungen des Abschnitts 6.2 und stellen für das jeweilige Beispiel das dazugehörige Flußdiagramm vor.

Bei diesen Flußdiagrammen werden wir eine abkürzende Schreibweise wählen. Anstatt zu schreiben :

```
Belege ein Feld
I mit dem Wert 1
```

schreiben wir einfach :

```
I = 1
```

weil die knappere Darstellung einfach überschaubarer ist.

5. Programm :

Nach der Problemanalyse des stellen wir dann das jeweilige Programm vor.

An dieser Stelle ist der Hinweis wichtig, daß es zur Lösung eines bestimmten Problems nicht nur ein Programm gibt. Es sind in der Regel unterschiedliche Programmversionen und -varianten vorstellbar, die alle das gleiche Problem lösen können.

Wenn also bei einer bestimmten Fragestellung der Leser vielleicht anders programmiert hätte als es hier vorgestellt wird, ist dies nicht weiter verwunderlich. Entscheidend ist, daß das jeweilige Programm die korrekte Problemlösung bereitstellt.

6. Variablenliste :

Zur vollständigen Dokumentation des jeweiligen Programms fügen wir im folgenden Abschnitt eine komplette alphabetische Variablenliste an, um dem Leser die Lektüre der Programme zu erleichtern.

7. Programmbeschreibung :

Ebenfalls aus Gründen der Dokumentation und der Erleichterung des Nachvollzugs der Programme fügen wir im siebten Abschnitt eine Programmbeschreibung an, die Satz für Satz erläutert, was in dem jeweiliegn Programm geschieht. Für den etwas ungeübteren Leser ist dieser Abschnitt sicherlich einer der wichtigsten.

8. Programmergebnisse :

In diesem Abschnitt werden, beispielsweise anhand von "Spieldaten", die Programmergebnisse vorgestellt. Auch dies ist eine Methode, um das Verstehen eines Progamms und den Nachvollzug der einzelnen Programmschritte zu erleichtern.

9. Ausblick :

Im letzten Abschnitt des jeweiligen Kapitels stellen wir dann einige Informationen bereit, die es dem Leser ermöglichen, abzuschätzen, wie das jeweilige Problem in einen größeren Gesamtzusammenhang eingeordnet werden kann.

Kapitel 7 : Expertensystem

7.1 Aufgabenstellung

Worum geht es, wenn man über Expertensysteme spricht ?

Der Name derartiger Programme und Programmsysteme verdeutlicht, daß versucht wird, mit Hilfe von Computern Experten quasi zu ersetzen, die über bestimmte Problembereiche mehr wissen als wir selbst.

Man kann sich auch vorstellen, daß das in einem Lexikon gespeicherte Wissen bei hinreichender Kapazität einer Rechenanlage in dieser gespeichert werden kann. Vielleicht können auch die wichtigsten Tricks, die ein Klempner kennt - wenn auch wohl nicht seine handwerklichen Fertigkeiten - in einem Rechner gespeichert werden, und auch die wesentlichen hilfreichen Hintergrundinformationen, die ein Arzt bei der Erstellung seiner Diagnose benutzt, können abrufbar gespeichert werden.

Sicherlich können wir mit einem noch so guten medizinisch-diagnostischen System nicht den guten Rat des Arztes ersetzen, aber dieser selbst kann seine Entscheidungen unter Nutzung eines solchen Systems verbessern. Man stelle sich zum Beispiel das folgende Problem vor :

Ein Patient kommt zum Arzt und klagt über Fieber und Schmerzen in der Brust. Diese Symptome treffen auf eine Vielzahl von durchaus verschiedenen Krankheiten zu, so daß der Arzt sicherlich erst dann eine Diagnose wagen kann, wenn er weitere Symptome feststellen oder erfahren kann. Aber selbst wenn er seine Informationsbasis verbessern kann, wäre es durchaus denkbar, daß er nicht

entscheiden kann, welche von verschiedenen Krankheiten zutreffen könnte, bzw. es könnte der Fall eintreten, daß er zufälligerweise an ganz bestimmte Krankheiten im Moment gar nicht denkt.

Es wäre dem Arzt sicherlich eine Hilfe, wenn ein medizinisches Diagnosesystem alle Krankheiten und die dazugehörigen Symptome auflisten könnte, die - von seinem aktuellen Wissensstand ausgehend - in Frage kommen könnten.

Wenn ein derartiges System funktioniert, dann muß es auch umgekehrt in der Lage sein, auszugeben, welche Symptome auftreten müßten, wenn das Vorliegen einer bestimmten Krankheit einmal unterstellt wird.

Wir erkennen an diesem Beispiel sehr deutlich, wie man sich ein Expertensystem vorzustellen hat und es ist einzusehen, daß auch außerhalb des medizinischen Bereichs entsprechende Systeme etabliert werden können : Man denke etwa an die Geologie (Auswertung von Bodenproben oder an den militärischen Bereich (Auswertung von Satellitenaufnahmen oder von Agentenmeldungen).

Allerdings gehört zu einem Expertensystem noch mehr als nur die Bereitstellung einer "Datenbank", aus der im Bedarfsfall Informationen abgerufen werden können. Zu diesem Bereich der künstlichen Intelligenz zählt man auch diejenigen Aufgaben, die mit der Etablierung und der Pflege eines derartigen Systems zusammenhängen. Hinzu kommt, daß von einem leistungsfähigen Expertensystem auch erwartet wird, daß es seinerseits lernfähig ist, seine eigene Wissensbasis also im Lauf der Zeit selbsttätig ausweiten kann (über lernfähige Programme sprechen wir in einem späteren Kapitel).

Somit ist ein "echtes" Expertensystem ein umfangreiches Progammpaket, das aus mehreren unterschiedlichen, aber zusammenarbeitenden Komponenten zusammengesetzt wird. Diese Komponenten sind die folgenden :

1. Wissensbasis (Datenbasis)
2. Problemlösungsprogramm
3. Benutzerschnittstelle
(4. Bereich des "Selbstlernens")

Für diese einzelnen Bereiche verweisen wir auf die zuständige Spezialliteratur (z.B.: STEDE,M. u.a. : 1984, insbesondere Kapitel 1; STEINACKER,I. : 1984, Abschnitt 1.3.5; RETTI,J. : 1984).

In der zuletzt erwähnten Arbeit von RETTI findet sich eine detailliertere Komponentengliederung eines Expertensystems, die wir hier stichwortartig ebenfalls vorstellen, weil sie sehr illustrativ zeigt, worum es geht :

Komponenten eines Expertensystems :

- Wissensbasis (Datenbasis)
- Inferenzkomponente (Entscheidungsalgorithmus)
- Erklärungskomponente
- Wissenserwerbskomponente
- Dialogkomponente

(RETTI,J. : 1984, S.78-87).

Abschließend weisen wir darauf hin, daß häufig auch die Erstellung von Expertensystemen ("Knowledge Engineering") mit zum Thema "Expertensysteme" gezählt wird (siehe dazu z.B. STEDE,M. u.a., 1984, Seite 24 ff.).

7.2 Lösungsansätze

Aus den Ausführungen des vorangegangenen Abschnitts geht schon deutlich hervor, wie ein Computerprogramm, das als Expertensystem bezeichnet werden kann, aufgebaut werden müßte. Die einzelnen Arbeitsschritte, die erforderlich werden, sind die folgenden :

1. Sammlung aller zu einem bestimmten Fachgebiet vorliegenden Informationen.

2. Errichtung der sog. Datenbank, d.h. Eingabe dieser Informationen in eine Rechenanlage unter besonderer Beachtung der Zugriffsmöglichkeiten.

3. Entwicklung des Problemlösungsalgorithmus, d.h. Erstellung eines Computerprogramms, das den Rechner in die Lage versetzt, von den Anforderungen und Eingaben des Benutzers die zutreffenden Antworten bzw. Auskünfte zu finden (dazu gehört auch die geschickte Programmierung von Ein- und Ausgabe).

4. Entwicklung des Programmteils, welcher für die selbsttätige Erweiterung der Wissensbasis in dem Fall zuständig ist, wenn ein selbstlernendes Expertensystem etabliert werden soll.

Bei der Lösung dieser Teilaufgaben spielen insbesondere die kapazitären Möglichkeiten der zur Verfügung stehenden Rechenanlage und auch die Geschwindigkeit des Zugriffs auf die einzelnen Informationen der Datenbasis, beziehungsweise die Geschwindigkeit des Ablaufs der programmierten Entscheidungsprozesse die entscheidende Rolle. Einem Kleinrechner wie dem Commodore C 64 sind diesbezüglich natürlich sehr enge Grenzen gesetzt.

Auch wenn der Grundgedanke der zu leistenden Programmierarbeit eigentlich sehr einfach ist und sich mit schlichten Wenn ... dann - Bedingungen

beschreiben läßt, kann doch die Detailarbeit sehr kompliziert werden. Wir werden dies in den nun folgenden Abschnitten zeigen, wobei der oben genannte Punkt 4 (Algorithmus des selbsttätigen Wissenserwerbs) in einem späteren Kapitel besprochen wird.

7.3 Das Beispiel

Ein "echtes" Expertensystem ist notwendigerweise sehr umfangreich. Deshalb sind - wie schon erwähnt - Kleinrechner nicht sehr gut dafür geeignet, solche Systeme zu etablieren. Aber selbst ein für Kleinrechner geeignetes System, wie zum Beispiel das Expertensystem XPER von DATA-BECKER kann wegen seiner Komplexität hier nicht dargestellt werden, wenn der Leser die Möglichkeit bekommen soll, nachzuvollziehen, wie ein derartiges System im Prinzip funktioniert.

Trotzdem kann anhand eines Demonstrationsbeipiels über Expertensysteme gesprochen werden, wenn wir uns auf die Simulation eines derartigen Systems beschränken. In Anlehnung an die medizinischen Expertensysteme, soll auch in diesem simulierten System eine "Diagnose" aufgrund der vom Benutzer des Systems einzugebenden Befunde gestellt werden.

Diese "Befunde" seien sozio - ökonomische statistische Angaben für Länder dieser Erde und die "Diagnose" besteht dann darin, daß das Expertensystem dem Benutzer sagt, um welches Land oder um welche Länder es sich handeln kann.

So wie der Arzt die medizinischen Befunde (Körpertemparatur, Blutdruckwerte, Körpergewicht, frühere Krankheiten des Patienten, Beschwerden etc.) eingibt und vom Rechner erfährt, um welche Krankheiten es sich auf der Grundlage dieser Informationslage handeln könnte, so geben wir sozio-ökonomische Befunde ein und erfahren dann,

auf welche Länder diese Befunde zutreffen.

Die sozio-ökonomischen Variablen, die wir in diesem Beispiel verwenden, sind die folgenden :

- Bevölkerungszahl
- Bruttosozialprodukt je Kopf (BSP)
- Inflationsrate
- Alphabetisierungsquote
- Lebenserwartung
- Bevölkerungsdichte
- Militärausgaben
- Bevölkerungszuwachs pro Jahr

Diese Aufzählung ist relativ willkürlich. Sie bezieht sich zudem nur auf quantitative Angaben.

Daraus ergibt sich, daß diese Liste sehr wohl verlängert und insbesondere auch um qualitative Angaben ergänzt werden kann (zum Beispiel Art des Regierungssystems, Typ des Gesellschaftssystems, Kontinent u.ä.).

Es kommt hier aber nicht darauf an, eine komplette und inhaltlich besonders aussagekräftige Indikatorenliste zu erstellen, sondern es soll ja nur die prinzipielle Vorgehensweise illustriert werden. Diese Vorgehensweise ist nicht davon abhängig, welche Indikatoren verwendet werden, sondern nur davon, daß überhaupt Indikatoren eingesetzt werden. Deshalb wollen wir uns mit diesen inhaltlichen Fragen nicht zu viel Mühe geben und uns auf die prinzipielle Vorgehensweise konzentrieren.

In diesem Beispiel geht es nun darum, daß der Benutzer Angaben bezüglich der oben genannten sozio-ökonomischen Variablen macht (sofern er über die betreffenden Informationen überhaupt verfügt, also gewissermaßen "Symptome" benennen kann), und der Rechner ihm daraufhin angibt, welche Länder zu diesem "Befund" passen.

Hier ist auf eine weitere Beschränkung in diesem Beispiel hinzuweisen, die daraus resultiert, daß

wir nur eine bestimmte Anzahl von Ländern in die Länderliste des folgenden Programms aufgenommen haben, um es nicht zu umfangreich werden zu lassen.

Der Leser wird beim Studium der Programmliste sofort erkennen, an welchen Stellen das Programm verlängert werden muß, wenn er die Länderliste ausweiten möchte. Allerdings würde es eine derartige Ausweitung erforderlich machen, daß er dann auch für die neu aufgenommenen Länder die entsprechenden Daten für die oben genannten sozio-ökonomischen Variablen bereitstellt. Die dafür zuständigen DATA-Statements wird er ebenfalls ohne Schwierigkeiten der Programmliste entnehmen können (auch die READ-Anweisungen müssen natürlich verändert werden, wenn mehr Länder und/oder mehr Variablen berücksichtigt werden sollen). Die statistischen Angaben für weitere Länder findet man in den Statistischen Jahrbüchern der Vereinten Nationen oder in den statistischen Veröffentlichungen der einzelnen Länder.

Wir gehen hier also von einem beschränkten Datenbestand aus, was für den Zweck, der mit diesem Simulationsbeispiel angestrebt wird, auch ausreicht.

Diesem Simulationscharakter entspricht es, daß der Benutzer keine exakten Werte für die einzelnen Variablen eingibt, sondern nur Näherungswerte oder sogar nur Schätzungen oder Vermutungen. Wenn er nämlich exakte Werte eingeben könnte, dann müßten wir unterstellen, daß er genau weiß, um welches Land es sich handelt - der Rat des Experten Computer wäre dann sicherlich überflüssig. Die Aufgabe unseres Expertensystems, "Diagnosen" zu stellen, ist nur dann sinnvoll, wenn aufgrund unvollständiger, unpräziser oder vorläufiger Informationen ("Befunde") das System ein schlußfolgerndes Ergebnis bereitstellen soll.

Deshalb muß dieses Simulationsbeispiel so angelegt werden, daß es auf der Grundlage von ungefähren Schätzbereichen für die einzelnen Variablen zu

verwertbaren Ergebnissen gelangen kann.

Wir geben also als Befund nicht ein :

Bevölkerungszahl 1982 beträgt 16 721 314

(dies könnte die exakte Bevölkerungszahl der DDR zu einem bestimmten Stichtag gewesen sein), sondern wir geben an :

Bevölkerungszahl 1982 ca. 17 Millionen.

Schon auf dieser ungenauen Grundlage sollte das simulierte Expertensystem in der Lage sein, festzustellen, daß als Land die DDR in Frage kommen könnte. Allerdings wird es auch noch andere Länder nennen können (und soll das natürlich auch), die diese oder eine ähnlich hohe Bevölkerungszahl aufweisen.

Erst wenn weitere Informationen eingegeben werden, wird das System in der Lage sein, die zunächst ausgegebene Länderliste zu verkürzen und schließlich vielleicht nur noch ein einziges Land ausgeben. Dem entspricht es, daß beispielsweise ein medizinisches Informationssystem bei der Angabe "38.5 Grad Fieber" sehr viele verschiedene Krankheiten als Diagnose ausgeben kann und erst, wenn weitere Informationen eingegeben werden, zu einer Einschränkung gelangen kann (wenn überhaupt).

Wir hatten schon darauf hingewiesen, daß ein Expertensystem auch für Auskünfte zuständig sein sollte, die quasi "umgekehrt" laufen : Gibt man beispielsweise ein bestimmtes Land ein (im Medizin-Beispiel eine bestimmte Krankheit), dann sollte das System die Werte der sozio-ökonomischen Variablen ausgeben, die für dieses Land zutreffen (im Medizin-Beispiel die zu erwartenden Befunde).

Diese "umgedrehte" Verwendungsrichtung werden wir im folgenden Kapitel vorstellen.

7.4 Problemanalyse

Bei der Vorbereitung der Erstellung des BASIC-Programms, welches ein Expertensystem gemäß der Beschreibung im vorangegangenen Abschnitt simulieren soll, kommt, wie bei allen folgenden Problemen auch, der Problemanalyse entscheidende Bedeutung zu :

Wir orientieren uns dabei am sog. EVA-Prinzip, d.h. wir überlegen also zunächst, welche Eingabe-Informationen das Programm benötigt, welche Verarbeitungsschritte folgen müssen und schließlich, welche Ausgabe-Informationen bereitgestellt werden sollen.

Der Eingabeteil des Programms ist recht einfach : Das Programm benötigt eine Länderliste und die konkreten Zahlenwerte für die quantitativen Variablen, mit deren Hilfe wir die einzelnen Länder beschreiben können. Zusätzlich können wir auch die Variablenbezeichnungen eingeben, um die späteren Ausgaben lesbarer zu gestalten.

Im Verarbeitungsschritt sind die folgenden Teilaufgaben zu bewältigen :

1. Eingabe eines Befundes, also eines Wertes für die erste der sozio-ökonomischen Variablen;

2. Ausgabe derjenigen Länder, für die dieser "Befund" zutrifft.

3. Prüfung, ob weitere "Befunde" eingegeben werden können, d.h. ob für das gesuchte Land noch weitere Informationen vorliegen.

4. Ist dies der Fall, so ist auch die nächste Information einzugeben.

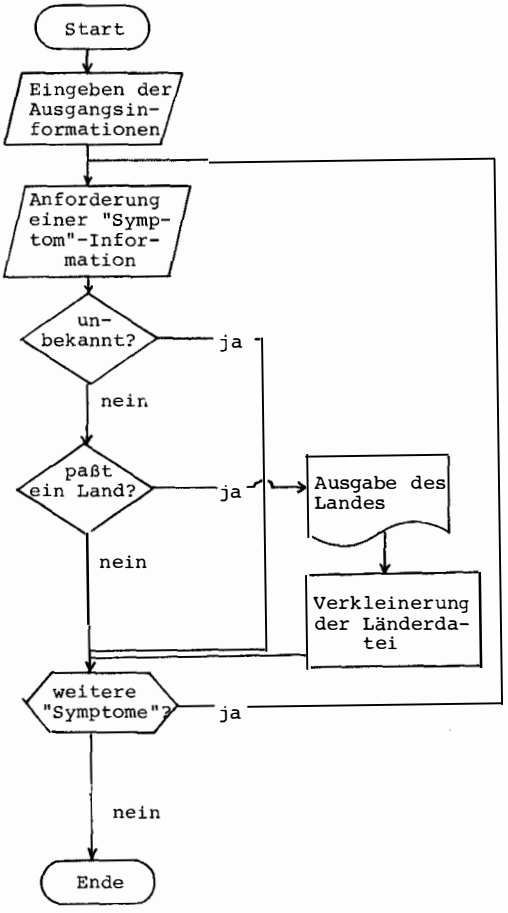
5. Aus der in Punkt 2. verkürzten Länderliste sind nun diejenigen Länder auszuwählen, für die auch

diese nächste Information noch zutrifft.

6. Weiter bei Punkt 3.

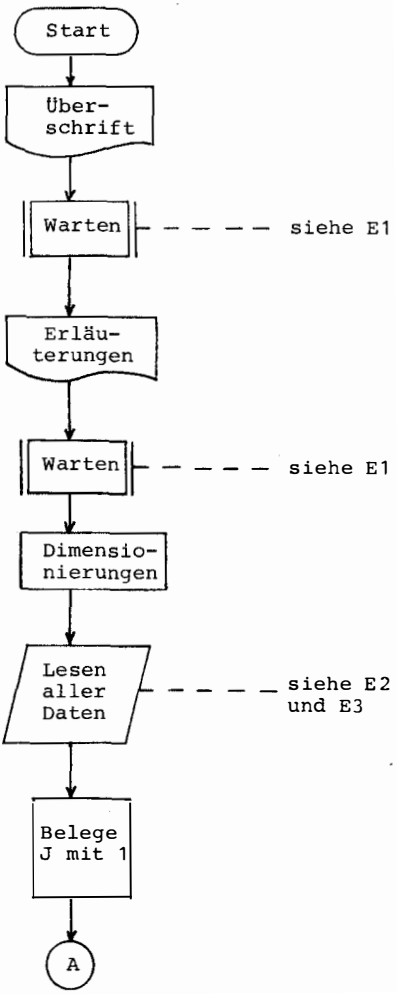
Damit gelangen wir zu dem folgenden generellen Programmablaufplan :

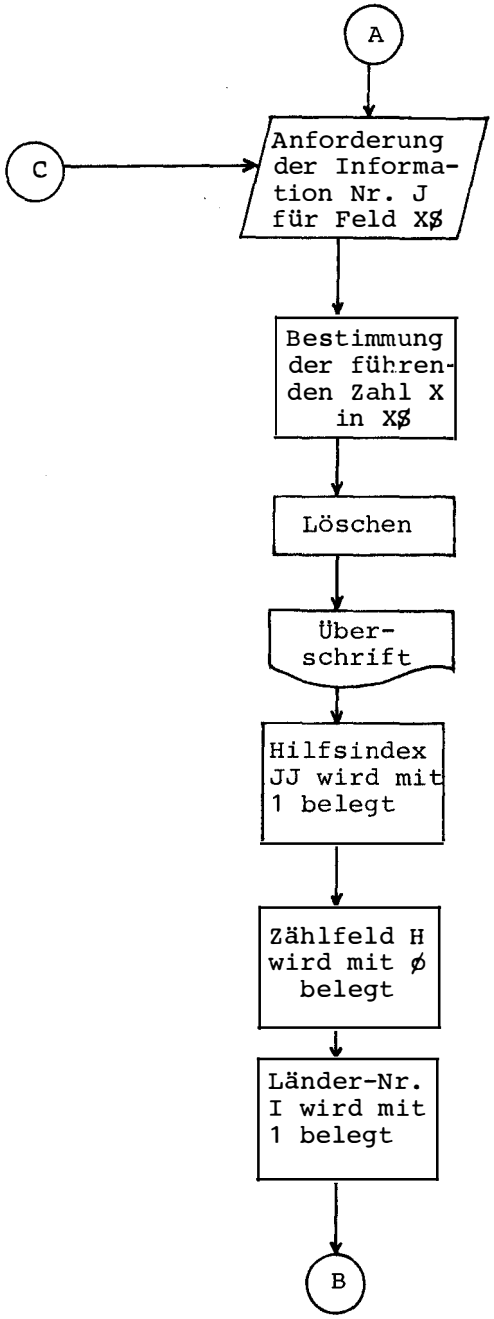
Flußdiagramm Expertensystem :



Wenn man dieses Flußdiagramm nun weiter aufgliedert, um die einzelnen Programmschritte zu erkennen, dann gelangt man zu der folgenden Darstellung :

Flußdiagramm : Detailversion





1.

2.

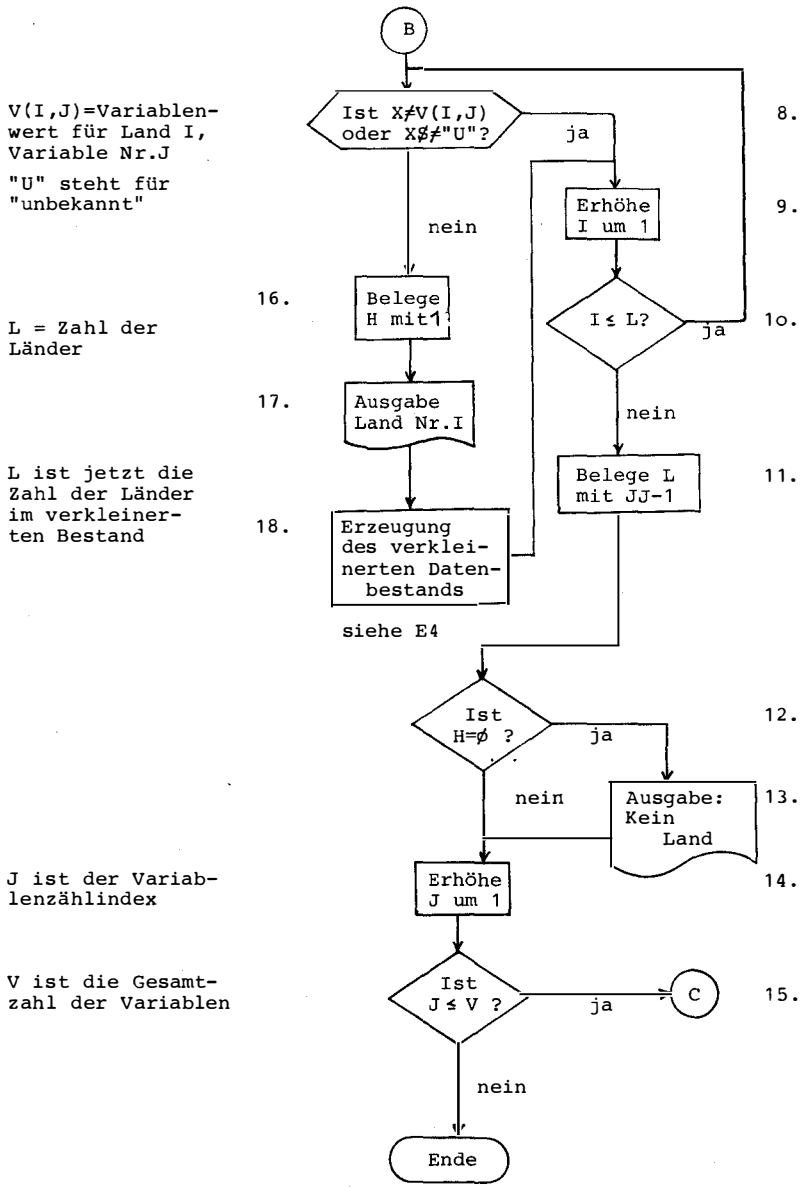
3.

4.

5. (notwendig, um die Länderdatei verkleinern zu können)

6.

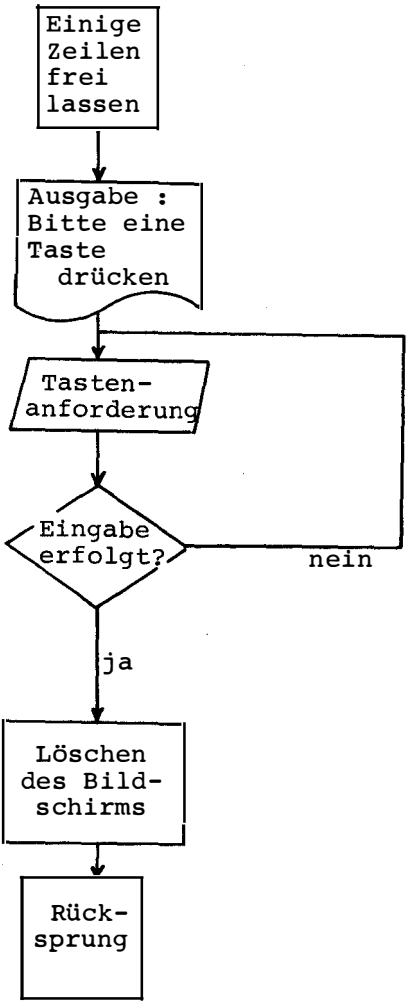
7.


B
V(I,J)=Variablenwert für Land I, Variable Nr.J
"U" steht für "unbekannt"
Ist X≠V(I,J) oder X$≠"U"?
ja
8.
Erhöhe I um 1
9.
nein
16.
Belege H mit1
L = Zahl der Länder
I ≤ L?
ja
1o.
17.
Ausgabe Land Nr.I
nein
L ist jetzt die Zahl der Länder im verkleinerten Bestand
Belege L mit JJ-1
11.
18.
Erzeugung des verkleinerten Datenbestands
siehe E4
Ist H=ø ?
12.
ja
nein
Ausgabe: Kein Land
13.
J ist der Variablenzählindex
Erhöhe J um 1
14.
V ist die Gesamtzahl der Variablen
Ist J ≤ V ?
ja
C
15.
nein
Ende

Auch in dieser Detailbetrachtung finden sich noch einige Bereiche, die in diesem ersten Demonstrationsbeispiel einer präziseren Betrachtung bedürfen.

Dazu gehört das in den meisten der folgenden Programme auch auftauchenden Unterprogramm zum Abwarten einer Tastatureingabe :

E1 : Unterprogramm Warten

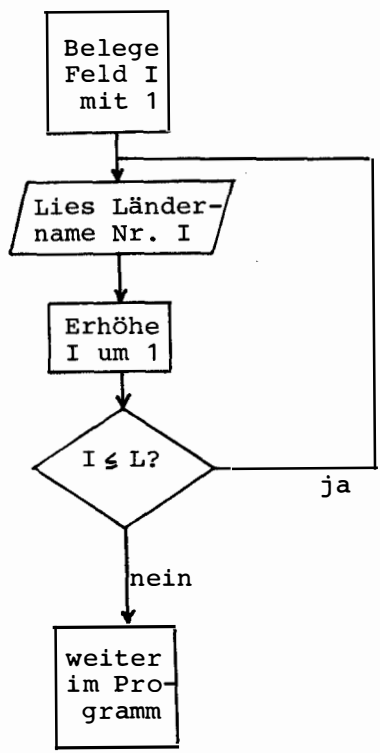


Auch der Programmteil, in dem die Ländernamen gelesen werden, soll in diesem ersten Beispiel gesondert betrachtet werden :

E2 : Einlesen Ländernamen

I ist der Länderzählindex

L ist die Zahl der Länder



Eine Detailbetrachtung widmen wir (auch nur in diesem ersten Beispiel) dem Einlesen der Daten :

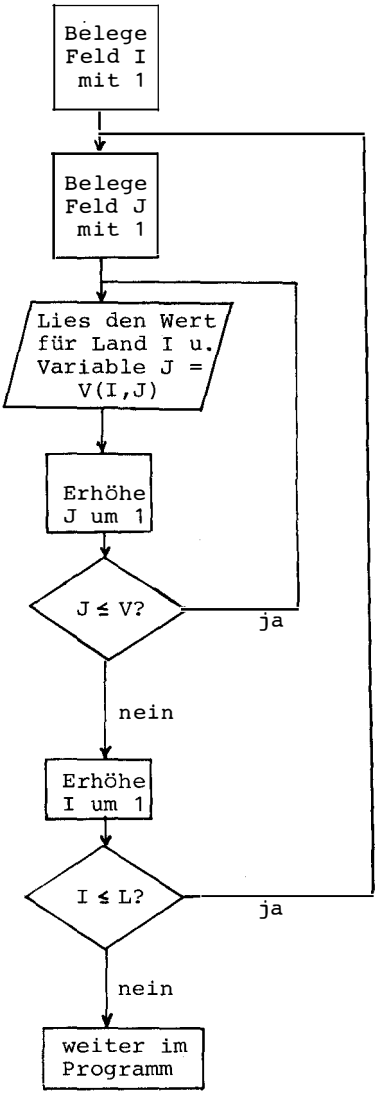
E3 : Einlesen Daten

I ist der Länderzählindex

J ist der Variablenzählindex

V ist die Zahl der Variablen

L ist die Zahl der Länder



Schließlich soll auch derjenige Programmschritt, der der Verkleinerung der Länderdatei dient, gesondert betrachtet werden :

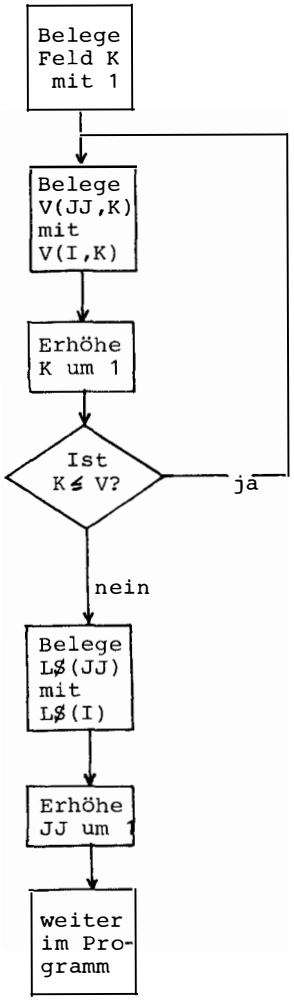
E4 : Verkleinerung der Länderdatei

K ist der Länderzählindex im verkleinerten Bestand

V bezeichnet die Variablenausprägungen

V ist die Zahl der Variablen

L$ bezeichnet die Ländernamen



1.
2.
3.
4.
5.
6.

Das Durcharbeiten dieser Flußdiagramme, die zusammengenommen das gesamte System unseres Simulationsprogramms hinreichend beschreiben, dürfte keine besonderen Schwierigkeiten bereiten, wenn man von dem etwas komplizierteren Mittelteil einmal absieht, wo es um die Anforderung von "Befunden" und die "Diagnosen" des Systems geht. Die einzelnen Schritte, die in diesem Bereich wichtig sind, sind die folgenden (die Numerierung orientiert sich an den Kennziffern im Detail - Flußdiagramm) :

1. Anforderung eines Befundes :

Dieser Befund wird als String im Speicherfeld X$ gespeichert. Falls der entsprechende Befund nicht vorliegt, so ist der Buchstabe "U" ("U" steht für "unbekannt") einzugeben.

2. Abspalten des numerischen Teils des Befundes :

Wir haben schon weiter oben darauf hingewiesen, daß in diesem Simulationsbeispiel nur quantitative (numerische) Größen beachtet werden. Gleichwohl erlaubt das Feld X$ auch die Stringeingabe, so daß das Programm bei Bedarf auch solche Informationen verarbeiten kann.

Dieser numerische Befund wird im Speicherfeld X gespeichert.

3. Löschen des Bildschirms :

Vor der Ausgabe der ersten Ergebnisse wird der Bildschirm "geräumt".

4. Ausgabe einer Überschrift für die Ergebnisse.

5. Belegung eines Hilfsfeldes JJ mit 1 :

Dieses Hilfsfeld JJ wird benötigt, um festzuhalten, wieviele Länder nach dem ersten "Befund" als Diagnose ausgegeben wurden, bzw.

wieviele demnach für die Kontrolle nach Eingabe des nächsten "Befundes" noch zur Verfügung stehen.

Der Grundgedanke dabei ist, daß nach Eingabe einer bestimmten Information nur noch diejenigen Länder zur Debatte stehen, die durch die vorhergehende Information ausgewählt wurden.

Dies bedeutet, daß der Länderbestand ständig verkleinert wird, je mehr Informationen der Benutzer bereitstellen kann.

6. Belegung eines Hilfsfeldes H mit 0 :

Dieses Hilfsfeld wird benötigt, um im weiteren Programmlauf feststellen zu können, ob eine bestimmte Information vom Benutzer eingegeben wurde, oder ob er bei der betreffenden Variablen "unbekannt" (= "U") geantwortet hat.

7. Belegung des Indexfeldes I mit 1 :

Mit dem Indexfeld I wird die Nummer des gerade betrachteten Landes angegeben. Wird diese Position erreicht, heißt dies also, daß das erste Land im Hinblick auf den vom Benutzer eingegebenen "Befund" geprüft wird, ob es diesem entspricht oder nicht.

8. Prüfung :

In dieser Position wird überprüft, ob der eingegebene Befund (X bzw. X$) dem entsprechenden Variablenwert des gerade betrachteten Landes (V(I,J)) entspricht.

Ist dies nicht der Fall, gelangen wir zur Position 9., andernfalls zur Position 16.

9. Erhöhung von I um 1 :

Da die Prüfung negativ ausgefallen ist, ist jetzt das nächste Land zu betrachten.

10. Prüfung, ob noch ein Land vorhanden :

Wenn noch ein Land vorhanden ist, springt das Programm zurück zur Position 8. (siehe dort); wenn nicht, geht es weiter bei Position 11.

11. Belegung des Hilfsfeldes L :

Die Position 11. wird nur erreicht, wenn kein Land mehr vorhanden ist. Es muß dann vor der Eingabe und Überprüfung der nächsten Inputinformation der Länderbestand verkleinert werden. In der nächsten Runde stehen nur noch diejenigen Länder zur Verfügung, die in der ersten Runde ausgewählt wurden. Es handelt sich um JJ-1 Länder, wie aus dem Teilprogramm E4 hervorgeht (der Laufindex JJ wird jedesmal erhöht, wenn bei der Prüfung in Position 8. ein Land als zutreffend erkannt wurde).

12. Prüfung für die Ergebnisausgabe :

Wenn die Position 12. erreicht wurde, ohne daß die Hilfsvariable H (siehe Position 6. und 16.) mit 1 belegt worden wäre, dann ist überhaupt kein Land gefunden worden, zu dem die vorher eingegebene Information paßt. Dieses Ergebnis wird in Position 13. ausgegeben; andernfalls geht es weiter bei Position 14.

13. Ausgabe, daß kein Land gefunden wurde.

14. Erhöhung von J um 1 :

Dies bedeutet, daß nun die nächste Information angefordert werden kann, sofern eine solche Informationen überhaupt noch vorliegt.

15. Prüfung, ob noch Informationen vorliegen :

Wenn noch Informationen vorliegen, geht es zurück zur Position 1. (siehe oben), wenn nicht, wird das Gesamtprogramm beendet.

16. Belegung des Hilfsfeldes H mit 1 :

Die Position 16. wird nur erreicht, wenn in Position 8. beim Vergleich der Länderangaben mit der Inputinformation Übereinstimmung festgestellt wurde. Dies bedeutet, daß ein Land gefunden wurde, das als Antwort des Expertensystems auszugeben ist. Dies geschieht in Position 17.

17. Ergebnisausgabe.

18. Verkleinerung der Länderliste :

Da ein "passendes" Land gefunden wurde, ist für weitere Prüfungen die Länderliste zu verkürzen. Dies geschieht im Teilprogramm E4.

Wenn dies geschehen ist, geht es weiter bei Position 9. (siehe dort), d.h. das nächste Land wird in die Prüfung aufgenommen.

Diese Ausführungen dürften hinreichend erläutern, was im zentralen Bereich des Programms geschieht, insbesondere wenn man diese Erläuterungen gemeinsam mit dem betreffenden Teil des Flußdiagramms und der Programmliste studiert.

Auch der Programmteil E4, der die Verkürzung der Länderliste (siehe oben : Position 18.) betrifft, soll im Detail betrachtet werden, weil er etwas komplizierter ist; auch dabei orientieren wir uns an den Positionsnummern des Flußdiagramms (E4) :

1. Belegung des Feldes K mit 1 :

K ist ein Laufindex, der die Nummer der jeweils zu verlagernden Variablen angibt; dieser Laufindex bewegt sich von 1 bis V (Anzahl der Variablen).

2. Belegung des Feldes V(JJ,K) mit V(I,K) :

Wenn dieser Programmteil zum ersten Mal berührt wird, dann deshalb, weil im Vergleich bei Schritt 8. der vorangegangenen Ablaufbeschreibung

festgestellt wurde, daß ein bestimmtes Land zu der eingegebenen Information "paßt". In diesem Fall hat die Variable JJ den Wert 1. Auch der Laufindex K hat den Wert 1 (siehe 1.), so daß Feld V(1,1) mit V(I,1) belegt wird.

Als erster neuer Variablenwert taucht also der erste Wert desjenigen Landes Nr. I auf, für das der Vergleich in Schritt 8. der vorangegangenen Beschreibung erfolgreich war. Dieses Land Nr. I steht dann als erstes Land in der nächsten Vergleichsrunde zur Verfügung.

3. Erhöhung von K um 1 :

Die gleiche Prozedur, wie sie im Punkt 2. beschrieben wurde, wird nun für die nächste Variable des gleichen Landes durchgeführt.

4. Prüfung, ob K kleiner oder gleich V :

Diese Verlagerungsprozesse werden für das Land Nr. I, das zum Land Nr. JJ (JJ = 1) der nächsten Vergleichsrunde wird, so lange durchgeführt, wie es Variablen zu verlagern gibt (also bis V = Anzahl der Variablen).

5. Belegung von L$(JJ) :

Wenn alle Variablen in die neue, nun verkürzte Datei übertragen worden sind (Schritte 1. bis 4.), dann wird auch der Name des Landes Nr. I (L$(I)) als neuer erster Name (L$(JJ); JJ = 1) gespeichert.

6. Erhöhung von JJ um 1 :

Dies ist die Vorbereitung für die (eventuell notwendige) Verlagerung eines weiteren Landes.

Damit ist auch diese Verlagerungsprozedur beschrieben, die erforderlich ist, um bei weiteren Informationseingaben in einer immer kleiner werdenden Länderliste zu untersuchen, welche

Länder als "Diagnose" nun noch in Frage kommen.

Wir haben in diesem Beispiel die Flußdiagramme sehr ausführlich dargestellt und erläutert, um einmal die Komplexität eines derartigen Programms zu veranschaulichen, und um den Leser mit unserem Stil der Lösung von Problemen etwas vertraut zu machen. Bei den folgenden Beispielen werden wir uns dann kürzer fassen können.

7.5 Programm

```
10 REM KI01 - EXPERTENSYSTEM
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,     1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 1 :":PRINT
60 PRINTTAB(8)"EXPERTENSYSTEM"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM SIMULIERT EIN SOG. EX-"
90 PRINT"PERTENSYSTEM :":PRINT
100 PRINT"NACH EINGABE VERWERTBARER INFORMATIONEN"
110 PRINT"UEBER SOZIO-OEKONOMISCHE SACHVERHALTE"
120 PRINT"VERSUCHT ES, DAS LAND ZU ERMITTELN, DAS"
130 PRINT"DURCH DIESE SACHVERHALTE BESCHRIEBEN"
140 PRINT"WIRD."
150 GOSUB 2000:REM WARTEN
160 REM EINLESEN DER DATEN
170 L=70
180 V=8
190 DIM L$(L),V$(V),V(L,V)
195 PRINTCHR$(147):PRINTTAB(12)"BITTE WARTEN"
200 FOR I=1 TO L:READ L$(I):NEXT I
210 FOR J=1 TO V:READ V$(J):NEXT J
220 FOR I=1 TO L:FOR J=1 TO V:READ V(I,J):NEXT J:NEXT I
230 REM ANFORDERUNG DER INFORMATIONEN
235 B$="BITTE INFORMATIONEN EINGEBEN :"
240 FOR J=1 TO V
242 IF J>1 THEN B$="BITTE WEITERE INFORMATIONEN :"
245 PRINT:PRINT:PRINT
250 PRINTTAB(3)B$
260 PRINTTAB(7)"(U, FALLS UNBEKANNT)":PRINT:PRINT
270 PRINTV$(J);:INPUT X$:X=VAL(X$)
272 PRINTCHR$(147)
274 PRINT"DIE FOLGENDEN LAENDER ENTSPRECHEN DEN"
276 PRINT"BISHER EINGEGEBENEN INFORMATIONEN :":PRINT:PRINT:PRINT
280 JJ=1
281 XP=.2*X
285 H=0
```

```
290 FOR I=1 TO L
300 IF ABS(X-V(I,J))>XP AND X$<>"U" THEN 360
305 H=1
310 PRINTTAB(12)L$(I)
320 REM ERZEUGUNG DES VERKLEINERTEN DATENBESTANDS
330 FOR K=1 TO V:V(JJ,K)=V(I,K):NEXT K
340 L$(JJ)=L$(I)
350 JJ=JJ+1
360 NEXT I
370 L=JJ-1
375 IF H=0 THEN PRINT:PRINT"DEN BEDINGUNGEN ENTSPRICHT
    KEIN LAND !":GOTO 390
380 NEXT J
390 PRINT:PRINT:PRINT:PRINT"ENDE DER AUSGABE":END
1000 REM DATEN
1010 DATA AFGHANISTAN,AEGYPTEN,ALBANIEN
1012 DATA ALGERIEN,ANGOLA,ARGENTINIEN
1014 DATA AETHIOPIEN,BANGLADESCH,BIRMA
1016 DATA BOLIVIEN,BRASILIEN,BR DEUTSCHLAND
1018 DATA CHILE,CHINA,COSTA RICA,DDR
1020 DATA DOMINIK.REPUBLIK,EL SALVADOR
1022 DATA ECUADOR,FRANKREICH,GUINEA
1024 DATA GHANA,GRIECHENLAND,GROSSBRITANNIEN
1026 DATA HAITI,HONGKONG,INDIEN,INDONESIEN
1028 DATA IRAK,IRAN,ISRAEL,JUGOSLAWIEN
1030 DATA JAPAN,KAMPUTSCHEA,KENIA,KOLUMBIEN
1032 DATA KOREA-NORD,KOREA-SUED,KUBA
1034 DATA KUWEIT,LAOS,LIBYEN,MALAYSIA
1036 DATA MALI,MEXIKO,MOSAMBIK,NICARAGUA
1038 DATA NIGERIA,OBERVOLTA,PAKISTAN,PERU
1040 DATA POLEN,PORTUGAL,SAUDI-ARABIEN
1042 DATA SENEGAL,SIMBABWE,SINGAPUR
1044 DATA SRI LANKA,SUEDAFRIKA,TANSANIA
1046 DATA THAILAND,TSCHAD,TUERKEI,UDSSR
1048 DATA UGANDA,USA,VER.ARAB.EMIRATE
1050 DATA VIETNAM,VENEZUELA,ZAIRE
1100 DATA BEVOELKERUNG       (MILL)
1102 DATA BSP JE KOPF        (US-$)
1104 DATA INFALTIONSRATE        (%)
1106 DATA ALPHABETENQUOTE       (%)
1108 DATA LEBENSERWARTUNG  (JAHRE)
1110 DATA BEV.DICHTE JE QKM
1112 DATA MILITAERAUSGABEN (MRD.$)
1114 DATA BEV.ZUWACHS           (%)
```

```
1200 DATA 15.9,240,-99,12,37,31,.2,2.3
1202 DATA 39.8,580,12,44,57,40,2.8,2.3
1204 DATA 2.7,740,-99,80,70,91,.2,2.7
1206 DATA 18.9,1870,13,35,56,8,.5,3.3
1208 DATA 7.1,470,21,10,42,5,-99,1.5
1210 DATA 27.7,2390,131,93,70,9,1.5,1.3
1212 DATA 31.1,140,4,15,40,25,.3,2.5
1214 DATA 88.5,130,17,26,46,588,-99,3
1216 DATA 34.8,170,11,70,54,48,-99,2.3
1218 DATA 5.6,570,22,63,50,6,.1,2.6
1220 DATA 118.7,2050,37,76,63,1,1.8,3.3
1222 DATA 60.9,13590,5,99,73,246,21.4,-2.8
1224 DATA 11.1,2150,186,90,67,14,.7,1.8
1226 DATA 976.7,290,-99,66,64,89,44,1.7
1228 DATA 2.2,1730,15,90,70,41,.02,2.8
1230 DATA 16.9,7180,-99,-99,72,-99,4.2,0.1
1232 DATA 5.4,1160,9,67,61,102,0.2,3.5
1234 DATA 4.5,660,11,62,63,196,.06,3.3
1236 DATA 8,1270,14,81,61,25,.2,3.2
1238 DATA 53.4,8270,12,98,-99,98,19,.8
1240 DATA 5.4,290,4,20,45,19,-99,2.5
1242 DATA 11.7,420,35,30,49,44,.1,2.9
1244 DATA 9.6,4380,14,88,74,71,2.1,1
1246 DATA 55.9,5030,20,98,74,229,-99,.4
1248 DATA 5,270,9,23,53,171,.01,-99
1250 DATA 5.1,4240,8,90,74,-99,.08,-99
1252 DATA 673.2,240,8,36,52,177,3.5,2.1
1254 DATA 146.6,430,21,62,53,75,1.7,2.6
1256 DATA 13.1,3020,14,75,56,27,2,4.2
1258 DATA 38.8,2180,20,50,59,20,5,3.1
1260 DATA 3.9,4500,40,-99,72,174,3,2
1262 DATA 22.3,2620,18,85,70,86,2,1
1264 DATA 115.3,6797,-99,99,-99,306,10,1
1266 DATA 7.9,70,-99,50,-99,44,-99,2.7
1268 DATA 15.9,420,11,50,55,24,.3,3.4
1270 DATA 26.7,1180,22,80,63,23,-99,3
1272 DATA 18.3,730,-99,90,65,134,1.2,2.8
1274 DATA 38.2,1520,20,93,65,372,2.6,2.3
1276 DATA 9.7,810,-99,96,73,84,1.1,2
```

```
1278 DATA 1.4,19830,18,60,70,62,1.1,4
1280 DATA 3.4,90,-99,41,43,15,-99,2.1
1282 DATA 3,8640,18,50,56,1,1.6,3.2
1284 DATA 13.9,1620,8,50,64,37,.5,2.7
1286 DATA 7,190,10,9,43,5,.03,2.6
1288 DATA 69.8,2090,19,81,65,34,.7,3.3
1290 DATA 12.1,230,11,28,47,12,-99,2.3
1292 DATA 2.6,740,13,90,56,18,.04,3
1294 DATA 84.7,1010,18,30,49,87,-99,2.5
1296 DATA 6.1,210,10,5,39,23,.03,2
1298 DATA 82.2,300,14,24,50,90,1,3.2
1300 DATA 17.4,930,31,80,58,13,.4,2.9
1302 DATA 35.3,3660,-99,100,-99,113,3.2,0.85
1304 DATA 9.8,2370,17,72,71,106,.7,.9
1306 DATA 9,11260,24,16,54,4,14.6,3
1308 DATA 5.7,450,8,10,43,26,.06,2.5
1310 DATA 7.4,630,9,74,55,18,.5,3.5
1312 DATA 2.4,4430,5,75,72,3927,.5,1.5
1314 DATA 14.7,270,13,85,66,213,.02,2.1
1316 DATA 29.3,2300,12,65,61,21,2.1,1.5
1318 DATA 18.7,280,12,66,52,18,.3,3.1
1320 DATA 47,670,10,84,63,88,.8,3.2
1322 DATA 4.5,120,8,15,41,3,.04,3.1
1324 DATA 44.9,1470,30,60,62,55,2.6,2.5
1336 DATA 262.4,3700,-99,99,-99,12,105.7,.92
1338 DATA 12.6,300,30,48,54,52,.1,3
1340 DATA 219.8,9700,14,95,70,23,110,.9
1342 DATA 1,26850,-99,56,63,8,1.2,3
1344 DATA 54.2,170,-99,87,63,145,-99,2.8
1346 DATA 14.9,3630,12,82,67,14,.6,2.9
1348 DATA 28.2,220,32,58,47,12,.3,2.6
2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE
     DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
```

7.6 Variablenliste

A$ = Inputvariable
(Antwort des Benutzers im UP 2000)

B$ = Stringvariable
(Informationen zur Eingabe von "Befunden")

H = Kontrollvariable
H=1, wenn ein passendes Land gefunden wurde, sonst H=0

I = Laufindex

J = Laufindex

JJ = Hilfsindex
(Erzeugung der verkürzten Datei)

K = Hilfsindex
(Variablenindex in der verkürzten Datei)

L = Zahl der Länder

L$() = Ländernamen

V = Zahl der Variablen

V(,) = Variablenwerte

V$() = Variablenbezeichnungen

X = VAL(X$)
(numerischer Teil der Eingabe durch den Benutzer)

XP = Hilfsfeld zur Aufnahme eines Prozentsatzes von X

X$ = Eingabe durch den Benutzer ("Befund")

7.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tastatureingabe

Satz 80-140 :

Erläuterungen zum Programm

Satz 150 :

wie Satz 70

Satz 160-195 :

Vorgabe der Zahl der Länder, der Zahl der Variablen, Dimensionierungen und Ausgabe einer Wartemeldung

Satz 200-210 :

Einlesen der Ländernamen und der Bezeichnungen der Variablen

Satz 220 :

Einlesen der Werte der Variablen

Satz 235 :

Belegung des Hilfsstrings B$ für die Anforderung von Informationen

Satz 240-242 :

Beginn der Anforderung von Informationen und ggf. Veränderung des Hilfsstrings (siehe Satz 235)

Satz 245-270 :

Anforderung einer Information und Abspaltung des numerischen Teils dieser Information

Satz 272-276 :

Vorbereitung der Ausgabe von Ergebnissen

Satz 280 :

Belegung des Hilfsfeldes JJ mit 1 (der Zählindex JJ erfaßt die Anzahl der ausgegebenen Länder)

Satz 281 :

Belegung des Hilfsfeldes XP mit dem fünften Teil des als Information eingegebenen Wertes X (vergl. Satz 270); mit dieser Vorgabe wird erreicht, daß alle Länder, die bezüglich der gerade betrachteten Variablen Werte aufweisen im Bereich X plus/minus 20%, als Ergebnisse bereitgestellt werden; dies entspricht der Überlegung, daß der Benutzer in der Regel nur ungefähre Werte als Informationen vorgeben kann; natürlich kann dieser Prozentsatz auch verändert werden, wenn der Benutzer des Programms dies für sinnvoll hält

Satz 285 :

Belegung der Hilfsgröße H mit dem Wert 0 : Diese Größe gibt an, ob überhaupt ein Land wenigstens zu der eingegebenen Information paßt; dann nämlich wird ihr der Wert 1 zugewiesen (Satz 305)

Satz 290 :

Beginn der Überprüfung, welches Land zu der eingegebenen Information paßt

Satz 300-305 :

Wenn der jeweilige Variablenwert um weiter als 20 % vom eingegebenen Wert abweicht (siehe Satz 281; diese Prozentvorgabe kann verändert werden), und wenn zugleich nicht "U" ("U" steht für "unbekannt") eingegeben wurde, dann erfolgt ein Sprung zum Satz 360, d.h. es wird das nächste Land überprüft. Ist diese Bedingung hingegen nicht erfüllt, geht es weiter bei Satz 305 (siehe auch Satz 285)

Satz 310 :

Ausgabe des gefundenen Landes

Satz 320-350 :

Erzeugung der verkleinerten Länderliste :

Wenn in Satz 310 ein Land als zur eingegebenen Information passend ausgegeben wurde, dann muß dieses Land auch für die folgenden Überprüfungen nach der Eingabe weiterer Informationen zur Verfügung stehen; Länder hingegen, die nicht passen, werden bei den folgenden Überprüfungen nicht berücksichtigt

Deshalb werden nun alle betroffenen Variablenwerte V(I,K) in den neuen Datensatz V(JJ,K) übertragen (Satz 330) und auch der Ländername wird neu gespeichert (Satz 340); danach wird der Hilfsindex JJ um 1 erhöht, um gegebenenfalls ein weiteres Land in diesen neuen Datenbestand aufnehmen zu können

Satz 360 :

Beendigung der Länder-Schleife (siehe Satz 290)

Satz 370 :

Festlegung der Anzahl der Länder in dem neuen, nun verkleinerten Datenbestand

Satz 375 :

Wird dieser Satz erreicht und H ist immer noch mit dem Wert 0 belegt (siehe Satz 285, bzw. Satz 305), dann bedeutet dies, daß kein Land gefunden wurde, welches zu der eingegebenen Information paßt; dies wird ausgegeben und danach zum Programmende verzweigt (Satz 390)

Satz 380 :

Ende der Eingabeschleife, d.h. Übergang zur Anforderung der nächsten Information zur weiteren Präzisierung der "Diagnose" (Rücksprung zum Satz 240)

Satz 390 :

Beendigung des Programms

Satz 1000-1050 :

Ländernamen in DATA-Statements

Satz 1100-1114 :

Bezeichnungen der Variablen in DATA-Statements

Satz 1200-1348 :

Werte der Variablen (länderweise) in DATA-Statements

Satz 2000-2040 :

Unterprogramm zum Abwarten einer Tastatureingabe :

Nach fünf Leerzeilen wird die Meldung

BITTE EINE TASTE DRUECKEN !

ausgegeben (Satz 2010-2020); diese Eingabe wird über Satz 2030 (GET-Statement) im Feld A$ gespeichert; solange in diesem Feld nichts steht,

solange also keine Taste berührt wurde, solange ist A$ = "" (d.h. in diesem Feld ist nichts drin)

Ist dies der Fall, wird zurückgesprungen an den Anfang des Satzes 2030, d.h. faktisch, daß das Programm wartet

Erst wenn eine Tastatureingabe erfolgt ist, wird der Satz 2040 erreicht, der ein Löschen des Bildschirms und den Rücksprung in das rufende Hauptprogramm ermöglicht

7.8 Programmergebnisse

Nach dem Starten des Programms erscheinen auf dem Bildschirm Überschrift und Erläuterungen und danach werden Informationen angefordert :

```
   BITTE INFORMATIONEN EINGEBEN  :
       (U, FALLS UNBEKANNT)

BEVOELKERUNG      (MILL.)  :  ?
```

Geben wir jetzt beispielsweise den Wert 17 ein, so meldet das System :

```
DIE FOLGENDEN LAENDER ENTSPRECHEN DEN
BISHER EINGEGEBENEN INFORMATIONEN :

          AFGHANISTAN
          ALGERIEN
          DDR
          KENIA
          KOREA-NORD
          MALAYSIA
          PERU
          SRI LANKA
          TANSANIA
```

VENEZUELA

BITTE WEITERE INFORMATIONEN :
(U, FALLS UNBEKANNT)

BSP JE KOPF (US-$) :

Geben wir jetzt beispielsweise den Wert 2000 ein, so erhalten wir die folgenden Ergebnisse :

DIE FOLGENDEN LAENDER ENTSPRECHEN DEN BISHER EINGEGEBENEN INFORMATIONEN :

ALGERIEN
MALAYSIA

BITTE WEITERE INFORMATIONEN :
(U, FALLS UNBEKANNT)

INFLATIONSRATE (%) ?

Geben wir jetzt den Wert 3 ein, so antwortet das System jetzt, daß diesen Bedingungen kein Land mehr entspricht. Die Programmausgabe wird damit beendet.

7.9 Ausblick

Ein Expertensystem, wie es mit dem obigen Programm ansatzweise simuliert wurde, kann unter vielen verschiedenen Gesichtspunkten verändert, ergänzt und verbessert werden.

Eine erste Ergänzung liegt auf der Hand, wenn man unterstellt, daß der Datenbestand, der das Informationssystem des Systems darstellt (Datenbank) größer ist als in obigem Beispiel. Dann wird es sich empfehlen, die Daten auf einem externen Speicher (Diskette) als Random - access - Datei abzulegen und vom laufenden Programm aus darauf (Land für Land) zuzugreifen.

Auf diese Weise wird erreicht, daß auch bei beschränkter Kapazität des Arbeitsspeichers des Rechners sehr große Informationsmengen als Hintergrundinformation bereitgestellt werden können.

Eine weitere Programmveränderung kann dahingehend vorgenommen werden, daß die Ergebnisse des Programmsystems, statt auf dem Bildschirm, auf einem angeschlossenen Drucker erscheinen.

Ergänzend ist darauf hinzuweisen, daß eine wesentliche Programmverbesserung darin besteht, die Ergebnisse des Systems mit numerischen Wahrscheinlichkeiten zu quantifizieren. Das Programmsystem gibt dann zum Beispiel nicht aus :

In Frage kommen Frankreich und England

sondern :

In Frage kommen

- Frankreich (Wahrscheinlichkeit 87.2 %)
- England (Wahrscheinlichkeit 63.4 %)

Wenn wir uns an das "Diagnosebeispiel" aus dem medizinischen Bereich erinnern, dann ist einsichtig, daß eine derartige Ausgabe den Informationswert des Expertensystems wesentlich verbessert.

Wie derartige verbesserte Ausgaben zustandekommen können, werden wir in einem späteren Kapitel besprechen.

Einleitend hatten wir schon darauf hingewiesen, daß ein derartiges System unter Umständen auch in "umgekehrter" Richtung verwendet werden kann : Statt also "Befunde" (hier Variablenwerte) einzugeben und "Diagnosen" (hier Länder) zu erhalten, können dann auch Länder eingegeben werden und das System liefert uns die zutreffenden Variablenwerte.

Dies entspricht der Vorgehensweise eines sog. Auskunftssystems, das also in ein Expertensystem integriert werden könnte. Wir wollen ein derartiges Auskunftssystem aber gesondert betrachten. Dazu dient das Programmbeispiel des folgenden Kapitels.

Kapitel 8 : Auskunftssystem

8.1 Aufgabenstellung

Schon im Zusammenhang mit der Erörterung von Expertensystemen ist klar geworden, was unter einem Auskunftssystem verstanden wird : Es geht darum, daß das Programm zu einem bestimmten, vom Benutzer einzugebenden Stichwort, alle dazu passenden relevanten Informationen ausgibt.

An einem einfachen Beispiel läßt sich diese Aufgabenstellung rasch verdeutlichen :

Das Stichwort sei beispielsweise ein Land dieser Erde, zum Beispiel das Stichwort Frankreich (der Leser erkennt, daß wir mit diesm Beispiel die Verbindung zu unserem simulierten Expertensystem des vorangegangenen Kapitels herstellen können). Das Programmsystem sollte nun in der Lage sein, uns möglichst viele Informationen zu diesem Stichwort zu liefern, also etwa demographische und/oder ökonomische Daten.

Es wäre denkbar, daß ausgegeben wird, wie groß die französische Bevölkerung ist, wie hoch heute das Bruttosozialprodukt ist, wie groß die Fläche des Landes und die geographische Lage sind, wie die Klimaverhältnisse, historische Daten usw.

Zur Illustration noch ein anderes Beispiel : Ein sehr einfaches Auskunftssystem könnte ein Programmsystem sein, welches für ein durch den Benutzer eingegebenes deutsches Wort den entsprechenden englischen (oder französischen, italienischen oder spanischen) Begriff ausgibt.

Dies wäre mit einem Programm möglich, das ein normales Taschenwörterbuch ersetzt - selbstverständlich kann dieses Programm als Auskunftsprogramm angesehen werden.

Nur am Rande weisen wir hier schon auf Probleme hin, die bei einem solchen Auskunftssystem gelöst werden müssen. Ein typisches Beispiel dafür ist die Mehrdeutigkeit mancher Begriffe, die auch durchaus anspruchsvolle Übersetzungsprogramme scheitern lassen können.

Der Leser kann sich sicherlich weitere Beispiele derartiger Auskunftsprogramme ausdenken : Nachschlagewerke irgendwelcher Art, Verzeichnisse von Telephonnummern, Postleitzahlen und Adressen, Archivierung von Schallplatten- oder Briefmarkensammlungen u.ä.

Es versteht sich, daß sie nicht nur mit Expertensystemen (im Sinne des Kapitel 7) verknüpft werden können, dessen "Umkehrung" sie gewissermaßen darstellen, sondern zum Beispiel auch mit Suchprogrammen , die wir im folgenden Kapitel ansprechen.

Eine derartige Verknüpfung empfiehlt sich spätestens dann, wenn die Informationsbasis so groß wird, daß sich ihre Auslagerung auf einen externen Speicher (Diskette) empfiehlt. Dann nämlich kommt es darauf an, daß die Suche nach dem in Frage stehenden Stichwort nicht zu lange dauert - und die wesentliche Verkürzung von Suchzeiten (Zugriffszeiten) wird durch "intelligente" Suchprozesse ermöglicht.

Natürlich ist ein Auskunftssystem um so leistungsfähiger, je besser und größer sein Informationshintergrund ist. Betrachten wir dazu noch einmal den "Wörterbuch-Ersatz" :

Wenn zu einem eingegebenen deutschen Wort alle die in Frage kommenden englischen Begriffe unterschiedlichen Bedeutungsinhalts ausgegeben werden können, dann hat der Benutzer am Bildschirm

(oder am Drucker) die Möglichkeit, denjenigen englischen Begriff aus der Vielzahl der Angebote auszuwählen, der in seinem jeweiligen Textzusammenhang angemessen ist.

Im "Frankreich-Beispiel" bedeutet dies, daß der Benutzer von einem guten, d.h. leistungsfähigen Auskunftssystem mehr interessante Informationen über Frankreich erhält als von einem weniger leistungsfähigen System.

Dem Aufbau der jeweiligen Informationsbasis muß also bei einem Auskunftssystem, wie übrigens natürlich auch schon bei einem Expertensystem, größte Aufmerksamkeit gewidmet werden.

8.2 Lösungsansätze

Der Aufbau eines Auskunftssystems muß ähnlichen Grundüberlegungen folgen, wie sie schon beim simulierten Expertensystem vorgestellt wurden. Generell gilt, daß das Programm in der Lage sein muß, zu einer eingegebenen Information aus seiner Datenbank (Informationshintergrund) zugehörige Informationen herauszusuchen und auszugeben. Wenn solche Informationen nicht vorliegen, ist die Ausgabe einer "Fehlanzeige" vorzusehen.

Am oben erwähnten Beispiel der Vokabelübersetzung läßt sich dieser prinzipielle Lösungsweg verdeutlichen :

Man stelle sich vor, in einen Rechner wären zehn Vokabelpaare deutsch-englisch eingegeben worden. Der Benutzer erwartet nach der Eingabe eines deutschen Wortes das entsprechende englische Wort oder eine Fehlanzeige, falls ein Wort eingegeben wurde, das bei den zehn Vokabelpaaren noch nicht vorhanden ist.

In diesem einfachen Fall verfügt also der Rechner

über eine Datenbank, die aus zwei Arrays der folgenden Struktur besteht :

D$ (deutsch)	E$ (englisch)
D$(1)	E$(1)
D$(2)	E$(2)
.....	.....
D$(10)	E$(10)

Wenn der Benutzer einen deutschen Begriff eingibt (zum Beispiel ins Feld X$), dann sucht das Programm im D$-Array so lange, bis eine Übereinstimmung zwischen X$ und D$(I) festgestellt wird. Der dazugehörige String im Feld E$(I) ist dann als Antwort des Programms auszugeben.

Wenn bei dieser Suche in keinem Fall eine Übereinstimmung zwischen X$ und D$(I) festgestellt wird, so ist die Fehlanzeige auszugeben.

Bei sehr großen Arrays ergibt sich allerdings ein unter Umständen sehr langer Suchprozeß. Es muß dann zusätzlich überlegt werden, wie diese Suche abgekürzt werden kann. Im Vokabelbeispiel könnte eine derartige Verkürzung nach alphabetischer Sortierung durch Einschränkung des Suchbereichs (Bildung von Teil-Arrays) zustandekommen.

Gehören zu einer Inputinformation mehrere Ausgabeinformationen (Mehrdeutigkeiten im Vokabelbeispiel, bzw. mehrere sozio-ökonomische Angaben im Länderbeispiel), so müssen dem D$-Array (in obigem Beispiel) mehrere Ausgabe-Arrays (und nicht nur einer, nämlich der E$-Array) gegenübergestellt werden. Bei erfolgreicher Suche müssen dann die Inhalte aller entsprechenden Felder (E$(I),F$(I),G$(I),... usw.) ausgegeben werden.

An der generellen Problematik und dem generellen Lösungsweg ändert sich dadurch aber nichts.

8.3 Das Beispiel

Das Beispiel, das wir ausgewählt haben, stellt einen Kompromiß zwischen dem oben erwähnten Wörterbuch-Ersatz und dem Länderbeispiel dar : Wir stellen ein Programm vor, welches für ein beliebiges chemisches Element , das vom Benutzer des Programms nach freier Wahl eingegeben werden kann, relevante Detailinformationen bereitstellt.

Die Hintergrundinformationen, die dieses Programm in seiner Datenbank benötigt, entnehmen wir dem Periodensystem der Elemente, wie es jedem Chemie-Schulbuch entnommen werden kann. Die Einzelinformationen, die wir bereitstellen, und die als "Auskünfte" des Programms ausgegeben werden können, sind die folgenden :

- chemisches Zeichen (Kürzel)
- Ordnungszahl im Periodensystem
- Atomgewicht
- Wichte
- Schmelzpunkt
- Siedepunkt
- Gruppenzugehörigkeit (I bis VIII,0)
- Nebengruppenzugehörigkeit (a,b)
- Elektronenschalen (K bis Q)

Dieses Beispiel findet sich auch in der Veröffentlichung "Das Schulbuch zum Commodore 64, DATA-BECKER, Düsseldorf 1984, Seite 122 ff.).

Es ist leicht einsichtig, daß noch mehr an Detailinformationen ausgegeben werden kann als oben genannt wurde. Beispielsweise könnten bestimmte Eigenschaften der einzelnen chemischen Elemente mit aufgeführt werden (z.B. elektrische

Leitfähigkeit) oder die englischen Bezeichnungen oder sonstige spezielle Charakteristika. Aus der Programmliste und der Progammbeschreibung geht deutlich hervor, an welcher Stelle des Progamms entsprechende Ergänzungen notwendig werden.

8.4 Problemanalyse

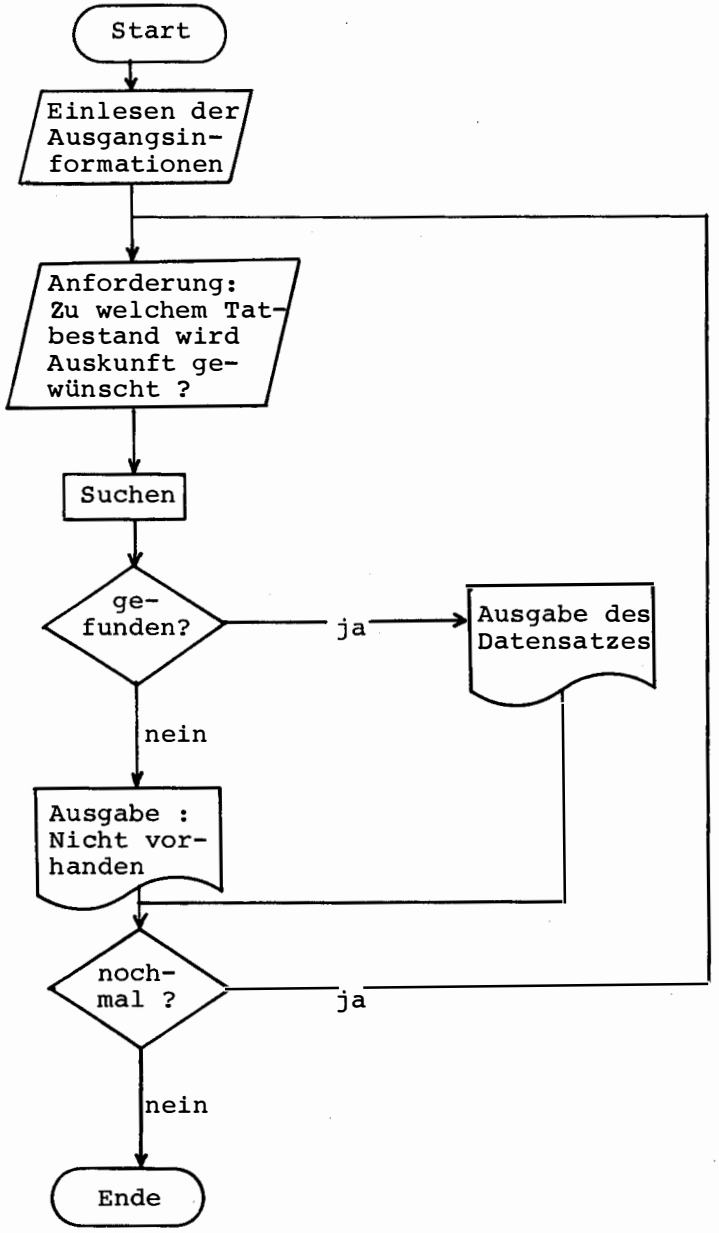
Die Hintergrundinformationen, die das Auskunftsprogramm benötigt, stellen wir über DATA- und READ-Statements zur Verfügung. Auch hier gilt, was auch schon im vorangegangenen Kapitel erwähnt wurde, daß größere Datenbestände sinnvollerweise auf einem externen Speicher (Diskette) abgelegt werden, um nicht bei jedem Programmstart den Einleseprozeß erforderlich zu machen, der bei größeren Datenbeständen vielleicht zu lange dauert.

Der Benutzer kann dann über ein INPUT-Statement dasjenige chemische Element eingeben, für das er Informationen wünscht und das Programm sucht dann die passenden Informationen in der Weise, wie in Abschnitt 8.2 beschrieben wurde.

In dem folgenden Programm sehen wir darüberhinaus vor, daß nach einer Auskunft auch eine weitere Auskunft bereitgestellt werden kann, wenn der Benutzer dies wünscht.

Somit gelangen wir zu dem folgenden generellen Programmablaufplan :

Flußdiagramm Auskunftsprogramm



Dieses Flußdiagramm ist so einfach, daß keine näheren Erläuterungen erforderlich sind.

8.5 Programm

```
10 REM KI02-AUSKUNFT
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ"
40 PRINT:PRINT:PRINT
45 PRINTTAB(5)"BEISPIEL 2  : AUSKUNFTSSYSTEM"
50 PRINT:PRINT:PRINTTAB(9)"PROF.DR.W.VOSS,  1985":PRINT
55 PRINT:PRINT:PRINT:PRINTTAB(13)"MOMENT BITTE":PRINT:PRINT
60 N=52:REM ANZAHL DER AUFGENOMMENEN ELEMENTE
70 DIM N$(N,10)
75 FOR J=1 TO 7
80 FOR I=1 TO N
85 READ N$(I,J)
90 NEXT I
95 NEXTJ
100 FOR I=1TON
105 READ N$(I,8),N$(I,9),N$(I,10)
108 NEXT I
110 PRINT:INPUT"WELCHES ELEMENT (NAME BITTE) ";B$
120 FOR I=1TON:IF B$=N$(I,1) THEN 160
130 NEXT I
140 PRINT:PRINT"DIESES ELEMENT IST IM DATENBESTAND"
150 PRINT"NICHT VORHANDEN.":GOTO 290
```

```
160 PRINTCHR$(147)
170 PRINT"■";N$(I,1);:PRINTTAB(23)N$(I,2)
180 PRINT"□":PRINT
190 PRINT"ORDNUNGSZAHL          : ";N$(I,3)
200 PRINT"GRUPPE                : ";N$(I,8)
210 PRINT"NEBENGRUPPE           : ";N$(I,9)
220 PRINT"SCHALE                : ";N$(I,10)
230 PRINT:PRINT
240 PRINT"ATOMGEWICHT           : ";N$(I,4)
250 PRINT"WICHTE                : ";N$(I,5)
260 PRINT
270 PRINT"SCHMELZPUNKT          : ";N$(I,6)
280 PRINT"SIEDEPUNKT            : ";N$(I,7)
290 PRINT:PRINT:INPUT"EIN ANDERES ELEMENT (J/N) ";A$
300 IF A$="J" THEN PRINTCHR$(147):GOTO 110
310 PRINT:PRINT"ENDE IN 310":END


495 REM DATEN BEGRIFFE
500 DATA ALUMINIUM,ANTIMON,ARGON,ARSEN,BARIUM,BLEI,BOR,BROM,CHLOR,CHROM
510 DATA EISEN,FLUOR,GOLD,HELIUM,IRIDIUM,JOD,KADMIUM,KALIUM,KALZIUM,KOBALT
520 DATA KOHLENSTOFF,KUPFER,LITHIUM,MAGNESIUM,MANGAN,MOLYBDAEN,NATRIUM
530 DATA NEON,NICKEL,PHOSPHOR,PLATIN,PLUTONIUM,QUECKSILBER,RADIUM,RADON
540 DATA SAUERSTOFF,SCHWEFEL,SELEN,SILBER,SILIZIUM,STICKSTOFF
550 DATA STRONTIUM,THORIUM,TITAN,URAN,VANADIUM,WASSERSTOFF,WISMUT,WOLFRAM
560 DATA XENON,ZINK,ZINN


595 REM DATEN ABKUEZUNGEN
600 DATA AL,SB,AR,AS,BA,PB,B,BR,CL,CR,FE,F,AU,HE,IR,J,CD,K,CA,CO,C,CU,LI,MG
610 DATA MN,MO,NA,NE,NI,P,PT,PU,HG,RA,RN,O,S,SE,AG,SI,N,SR,TH,TI,U,V,H
620 DATA BI,W,XE,ZN,SN


640 REM DATEN ORDNUNGSZAHLEN
650 DATA 13,51,18,33,56,82,5,35,17,24,26,9,79,2,77,53,48,19,20,27,6,29
660 DATA 3,12,25,42,11,10,28,15,78,94,80,88,86,8,16,34,47,14,7,38
670 DATA 90,22,92,23,1,83,74,54,30,50
```

```
695 REM DATEN ATOMGEWICHTE
700 DATA 26.97,121.76,39.944,74.91,137.36,207.21,10.82,79.919,35.457,52.01
710 DATA 55.85,19,197,4.003,192.2,126.92,112.41,39.096,40.08,58.94,12.011
720 DATA 63.54,6.94,24.32,54.94,95.95,22.991,20.183,58.71,30.98,195.09
730 DATA 242,200.61,226.05,222,16,32.066,78.96,107.88,28.06,14.008
740 DATA 87.63,232.05,47.9,238.07,50.95,1.008,209,183.86,131.3,65.38,118.7


745 REM DATEN WICHTE
750 DATA 2.7,6.67,.0018,5.72,3.5,11.34,1.73,3.14,1.557,7.14,7.86,.0017
760 DATA 19.25,.00018,22.42,4.942,8.64,.862,1.545,8.83,3.51D,8.933,.534
770 DATA 1.75,7.3,10.2,.971,.0009,8.8,1.83,21.4,-,13.55,-,.009,.0014,2.07
780 DATA 4.46,10.5,2.34,.00125,2.63,11.7,4.43,19,6.07,.0009,9.8,19.3
790 DATA .0059,7.14,7.28


795 REM DATEN SCHMELZPUNKTE
800 DATA 658,630,-190, ,704,327,2400,-7.3,-100,1800,1525,-218,1063,-272
810 DATA 2454,114,321,64,851,1490,-,1083,180,650,1221,2622,99
820 DATA -248.6,1452,44,1771,-,-39,700,-71,-218.7,113,220,960,1414
830 DATA -210,771,1827,1727,1689,1726,-262,271,3380,-111.9,419,232


845 REM DATEN SIEDEPUNKTE
850 DATA 2270,1440,-186,616,1537,1690,2550,58,-34,2600,2450,-187,2677
860 DATA -269,4400,184,770,762,1439,3185,-,2360,1372,1120,2152,4800,880
870 DATA -246,3075,280,3800,-,357,1140,-61.9,-182.97,444,688,1930,2630
880 DATA -195.8,1360,3530,-,-,3000,-272.78,1560,6000,-108.1,906,2430


900 REM DATEN PERIODISCHES SYSTEM
910 DATA III,B,M,V,B,O,0,A,M,V,B,N,II,A,P,IV,B,P,III,B,L,VII,B,N
920 DATA VII,B,M,VI,A,N,VIII,A,N,VII,B,L,I,B,P,0,A,K,VIII,A,P,VII,B,O
930 DATA II,B,O,I,A,N,II,A,N,VIII,A,N,IV,B,L,I,B,N,I,A,L,II,A,M,VII,A,N
940 DATA VI,A,O,I,A,M,0,A,L,VIII,A,N,V,B,M,VIII,A,P,-,-,-,II,B,P,II,A,Q,0,A,P
950 DATA VI,B,L,VI,B,M,VI,B,N,I,B,O,IV,B,M,V,B,L,II,A,O,-,-,-,IV,A,N
960 DATA -,-,-,V,A,N,I,A,K,V,B,P,VI,A,P,0,A,O,II,B,N,IV,B,O
999 END
```

8.6 Variablenliste

A$ = Antwortstring (Ja,Nein)

B$ = Stringvariable
(Eingabe des Benutzers)

I = Laufindex

J = Laufindex

N = Zahl der Elemente im Datenbestand

N$(,)= Dateielemente

8.7 Programmbeschreibung

Satz 10-50 :

Überschrift

Satz 55 :

Ausgabe einer Wartemeldung, weil das Einlesen der Hintergrundinformationen Zeit benötigt

Diese Meldung müßte modifiziert werden, wenn man sich entschließt, die Hintergrundinformationen auf einem externen Datenträger zu speichern, um die Einlesezeit zu verkürzen

Satz 60 :

Angabe der Anzahl der Elemente;

es sind in diesem Auskunftsprogramm nicht alle chemischen Elemente aufgenommen worden, sondern nur 52 Elemente; soll dieser Datenbestand

vergrößert werden, muß der Satz 60 verändert werden und es müssen selbstverständlich zusätzliche DATA-Statements angefügt werden (Satz 495 ff.)

Satz 70 :

Dimensionierung

Wenn mehr Detailinformationen pro Element bereitgestellt werden sollen, muß das zweite Argument in dieser Dimensionierungs-Anweisung vergrößert werden; auch hier gilt, daß dann zusätzliche DATA-Anweisungen erforderlich sind

Satz 75-95 :

Einlesen der Detailinformationen

Hier wird in einer geschachtelten Schleife das doppelt indizierte Feld N$(I,J) belegt, wobei sich der Laufindex J von 1 bis 7 (dies sind die ersten sieben Variablen pro Element) und der Laufindex I von 1 bis N (N = 52 = Zahl der im Datenbestand vorhandenen Elemente) bewegen

Satz 100-108 :

Einlesen der Variablenwerte für die Variablen Nr. 8, 9 und 10

Satz 110 :

Anforderung desjenigen Elements, über das der Benutzer nähere Auskünfte wünscht

Satz 120-130 :

Aufsuchen des gewünschten Elements im Datenbestand

Wird das Element gefunden, geht das Programm bei Satz 160 weiter; wird es hingegen nicht gefunden, geht es bei Satz 140 weiter

Satz 140-150 :

Ist das gewünschte Element im Datenbestand (noch) nicht vorhanden, so erfolgt eine entsprechende Meldung durch das Programm; danach weiter bei Satz 290

Satz 160 :

Löschen des Bildschirms vor den weiteren Ausgaben

Satz 170 :

Ausgabe des Namens des gewünschten Elements und des Kürzels in weißer Schrift

Satz 180 :

Umschalten auf die ursprüngliche Schriftfarbe und Ausgabe einer Leerzeile

Satz 190-280 :

Ausgabe der übrigen Informationen für das gewünschte Element (in gegenüber dem Einlesen veränderter Form)

Satz 290-300 :

Abfrage, ob noch eine weitere Auskunft über ein anderes chemisches Element gewünscht wird

Ist dies der Fall, wird der Bildschirm gelöscht und es erfolgt ein Programmsprung zum Satz 110; ist dies nicht der Fall, geht es weiter bei Satz 310

Satz 310 :

Beendigung des Programms

Satz 495-960 :

Datenbasis des Auskunftsprogramms in DATA-Statements :

es ist in diesem Zusammenhang zu beachten, daß die Variablen Nr.1 bis 7 (Begriffe, Abkürzungen, Ordnungszahlen, Atomgewichte, Wichten, Schmelzpunkte und Siedepunkte) jeweils einzeln für alle 52 Elemente angegeben sind (DATA-Statements Nr. 495-880);

bei den letzten drei Variablen hingegen (Gruppenzugehörigkeit, Nebengruppenzugehörigkeit, Elektronenschale) haben wir jeweils für ein Element alle drei Werte hintereinander angegeben (DATA-Statements Nr.900-960). Dies hat dann auch zu der unterschiedlichen Behandlung beim Einlesen geführt (der Leser betrachte noch einmal die Statements 100-108 im Vergleich mit den Statements 75-95)

8.8 Programmergebnisse

Wenn wir nach der Ausgabe der Programmüberschrift auf die Anforderung des Programms als Element, zu dem wir Auskünfte wünschen, beispielsweise das Element Schwefel eingeben, so antwortet das Programm :

```
SCHWEFEL                          S


ORDNUNGSZAHL                  :   16
GRUPPE                        :   VI
NEBENGRUPPE                   :   B
SCHALE                        :   M

ATOMGEWICHT                   :   32.066
WICHTE                        :   2.07

SCHMELZPUNKT                  :   113
SIEDEPUNKT                    :   444


EIN ANDERES ELEMENT (J/N)  ?
```

Geben wir auf diese Frage J (für Ja) ein, so wird erneut ein Elementname durch das Programm angefordert. Geben wir nicht J ein, so wird das Programm beendet und kann nur mit dem Kommando RUN erneut gestartet werden.

8.9 Ausblick

In diesem Abschnitt sind im Prinzip die gleichen Anmerkungen angebracht wie im entsprechenden Abschnitt des Kapitels zuvor, denn in der Tat bestehen ja eine ganze Reihe von Gemeinsamkeiten zwischen einem Expertensystem und einem sog. Auskunftssystem. Und nicht umsonst wird ja in der praktischen Nutzung ein Expertensystem häufig mit einem Auskunftssystem zu einem einzigen größeren Programmsystem zusammengefaßt.

In diesem Zusammenhang weisen wir darauf hin, daß insbesondere bei umfangreichen Hintergrund - Datenbeständen eine sog. Menü-Programmierung angebracht ist.

Bei dieser Art der Programmierung werden dem Benutzer Teilbereiche der Hintergrundinformationen zur Auswahl angeboten, vergleichbar einer Speisenkarte (Menü), aus der der Benutzer auswählen kann. Auf diese Weise wird er, unter Umständen in mehreren aufeinanderfolgenden Auswahlschritten, zu der gewünschten Auskunft hingeführt.

In unserem Programmbeispiel könnte eine solche Menüvorgabe beispielsweise so aussehen, daß der Benutzer vom Programmsystem zunächst gefragt wird, ob es sich um einen festen, um einen flüssigen oder um einen gasförmigen Stoff handelt, für den er Informationen wünscht.

Die Programmierung eines solchen Menüs ist mit Hilfe des ON ... GOTO-Statements recht einfach, wie im folgenden skizzenhaft gezeigt wird :

```
100 PRINT "HANDELT ES SICH UM EINEN":PRINT
102 PRINT "      FESTEN           (1)"
104 PRINT "      FLUESSIGEN       (2)"
106 PRINT "      GASFOERMIGEN     (3)":PRINT
108 PRINT "STOFF ?":PRINT
110 INPUT "BITTE NUMMER          :";Z
112 ON Z GOTO 200,300,400
 :
 :
200 REM PRUEFUNG FESTE STOFFE
 :
 :
300 REM PRUEFUNG FLUESSIGE STOFFE
 :
 :
400 REM PRUEFUNG GASE
 :
 :
```

Eine derartige Programmierung setzt natürlich voraus, daß der Datenbestand, der in den Hintergrundinformationen vorhanden ist, in entsprechender Weise strukturiert wird bzw. daß jedes Element im Datenbestand mit einem Kennwert versehen wird (z.B. 1 für "fest", 2 für "flüssig" und 3 für "gasförmig"; zu speichern in einer zusätzlichen Variablen), welcher im Zusammenhang mit dem ON ... GOTO-Statement zu überprüfen ist und die entsprechenden "Weichenstellungen" bewirkt.

Kapitel 9 : Suchprogramm

9.1 Aufgabenstellung

Im Zusammenhang mit dem Expertensystem aus Kap.7 und dem Auskunftsprogramm aus Kap.8 ist das Stichwort " Suchen " schon mehrfach gefallen. Es ging dabei jeweils darum, daß das Programmsystem in seinem Hintergrund-Datenbestand die jeweils benötigten Informationen suchen muß, die in Abhängigkeit von der Informationseingabe des Benutzers auszugeben sind.

Diese Suchprozesse dürfen nicht allzu viel Zeit verbrauchen, aber natürlich sind sie um so länger, je mehr Hintergrundinformationen bereitstehen.

Wir befinden uns also gewissermaßen in einer "Zwickmühle" : Ein Experten- oder Auskunftssystem ist um so leistungsfähiger, je umfangreicher sein Datenbestand ist; gleichzeitig wird es aber gerade deshalb weniger leistungsfähiger, weil die Suchprozesse, die erforderlich sind, immer länger dauern.

Man erkennt an dieser "Gegenläufigkeit" wie wichtig es ist, die benötigten Suchprozesse möglichst zu beschleunigen. Es geht im Prinzip darum, optimale Suchpfade zu finden.

Allerdings darf der Begriff des Suchens nicht nur im Zusammenhang mit dem Auffinden von wichtigen Informationen, die als Auskünfte ausgegeben werden sollen, interpretiert werden. "Suchen" kann auch im Rahmen des Computereinsatzes beispielsweise bei Spielprogrammen eine große Rolle spielen :

Man denke zum Beispiel an ein leistungsfähiges Schachprogramm : Die Spielzüge, die das Programm als Antwort auf Züge des Programmbenutzers durchführen muß mit der Zielsetzung, das Spiel zu gewinnen, müssen aus einer riesigen Zahl von denkbaren (erlaubten) Zug-Kombinationen ausgewählt werden. Insoweit ist die Auswahl einer ganz bestimmten Zugfolge aus der großen Zahl von alternativen Zug-Kombinationen nichts anderes als das Ergebnis eines zielgerichteten Suchprozesses.

Wir erkennen an diesem Beispiel, daß Suchen, genauer "optimales Suchen", interpretiert werden kann als die Festlegung einer Strategie, die eine bestimmte, vorher zu definierende Zielfunktion minimiert.

Häufig dürfte diese Zielfunktion einfach nur die Suchzeit sein. Man kann sich aber sehr wohl vorstellen, daß auch andere Zielfunktionen formuliert werden können. Ein Beispiel dafür sind etwa die Suchkosten , die minimiert werden sollen. Was nützt beispielsweise eine Halbierung der Suchzeit, wenn dies eine Vervierfachung der Suchkosten bedeutet ?

Wir wollen diese Fragen hier jedoch nicht weiter vertiefen. Sie werden unter den Stichworten "Minimierungsrechnung" und "Optimierungsrechnung" in der zuständigen Fachliteratur, insbeondere unter dem Gesichtspunkt der Herleitung geeigneter Methoden, behandelt (siehe zum Beispiel : VOß,W. : 1972).

Das Stichwort "Suchen" kann schließlich noch in einem ganz anderen Zusammenhang, wenn auch nach wie vor unter dem Gesichtspunkt des praktischen Computereinsatzes, verstanden werden : Wir denken hier beispielsweise an das Aufsuchen bestimmter Bildschirmpositionen im Zusammenhang mit der Lösung graphischer Aufgaben oder an geometrische Probleme, wie das Aufsuchen des Schnittpunkts von zwei Geraden und die Berechnung und Ausgabe der Koordinaten dieses Schnittpunkts.

In den Bereich derartiger Aufgaben gehören die mathematischen Probleme der Extremwert - und Wendepunktbestimmungen u.ä. An einem Beispiel aus diesem Bereich wollen wir das Schema eines Suchprogramms illustrieren.

9.2 Lösungsansätze

Soll ein Rechner mit Hilfe eines Programms eine zielorientierte Suche durchführen - unabhängig davon, um welche Art des Suchens es sich handelt - so ist es zunächst erforderlich, eine gegebene Zielfunktion zu formulieren.

Diesen Begriff der Zielfunktion kann man sich sehr anschaulich am Beispiel mathematischer Aufgaben klar machen :

In der Extremwertbestimmung (Auffinden des Maximums oder des Minimums einer mathematischen Funktion) geht es beispielsweise darum, den Wert (oder die Werte) der unabhängigen Variablen so zu bestimmen, daß die erste Ableitung für diesen Wert null wird.

Dies ist, wie der mathematisch vorgebildete Leser weiß, eine Aufgabe der Differentialrechnung . Nun gibt es allerdings auch mathematische Funktionen, die nur sehr schwer oder manchmal überhaupt nicht differenzierbar sind. In einem solchen Fall kann man das gesuchte Extremum dadurch finden, daß man alternative Werte der unabhängigen Variablen "durchprobiert", um stets durch Berechnung der jeweiligen Funktionswerte einen Bereich zu finden und einzugrenzen, in dem ein Extremum erwartet werden kann.

Dies ist dann ohne Zweifel ein Suchprozeß, der sehr einfach zu programmieren ist : Es ist lediglich dafür Sorge zu tragen, daß schrittweise der gleiche Rechenprozeß immer wiederholt wird,

bis ein Abbruchkriterium erfüllt ist, welches sinnvollerweise mit den Veränderungen der jeweiligen Zielfunktionswerte gekoppelt wird.

Somit wird deutlich, wie - programmlogisch gesehen - Suchen verstanden werden kann : In einer Folge sich wiederholender Arbeitsschritte wird die jeweilige Zielfunktion überprüft - mehr geschieht eigentlich nicht.

Problematisch wird die programmtechnische Lösung eines solchen Problems allerdings dann, wenn unterschiedliche Wege zum gleichen Ziel führen können, wenn also das Programm selbst eine Entscheidung darüber herbeiführen soll, welcher Suchpfad beschritten werden soll.

Ein einfaches Beispiel soll diese Problemstellung illustrieren :

Wir gehen einmal davon aus, daß im Datenbestand eines Rechners sich 1000 Namen befinden (zum Beispiel im Rahmen einer Adreßkartei). Es soll nun im Zusammenhang mit einem Auskunftsprogramm geprüft werden, ob ein ganz bestimmter Name (zum Beispiel der Name VOSS) in diesem Datenbestand vorhanden ist.

Dieses Überprüfen macht Suchprozesse erforderlich, die unterschiedlich strukturiert sein können. Zwei dieser verschiedenen Pfade wollen wir hier vorstellen :

1. Der einfachste Weg, besteht darin, den vorhandenen Datenbestand von Anfang an so lange zu durchsuchen, bis der gewünschte Begriff gefunden wird. Erreicht man bei dieser Suche erfolglos das Ende dieses Datenbestands, dann ist der gewünschte Begriff nicht vorhanden (genau so sind wir im Progammbeispiel des vorangegangenen Kapitels vorgegangen).

2. Ein anderer Suchpfad kann zum Beispiel so aussehen, daß wir in diesem Beispiel, in dem es um Namen geht, "alphabetisch" suchen. Dies bedeutet,

daß wir zunächst nur solche Namen untersuchen, die mit V anfangen; darunter dann nur solche, die an zweiter Stelle ein O stehen haben usw.

Im Einzelfall muß dann geprüft werden, welcher von verschiedenen Wegen der effizientere ist : Effizienzkriterien können - wie schon erwähnt wurde - die Suchzeit, die Suchkosten oder auch andere Kriterien oder Mischungen solcher Kriterien sein.

Im Computerbereich spielt das " binäre Suchen " eine sehr wichtige Rolle. Es geht dabei darum, jeden einzelnen Suchschritt so anzulegen, daß er als "Ja - Nein - Entscheidung" interpretiert werden kann. Dabei kommt es vor allem darauf an, die Folge der Suchschritte so aufzubauen, daß möglichst der effizienteste Suchpfad ausgewählt wird. Auch dazu wieder ein einfaches Illustrationsbeispiel :

Man stelle sich vor, der Computerbenutzer merkt sich eine Zahl (zum Beispiel 3456), die das ablaufende Computerprogramm "erraten" (suchen) soll. Dieses Programm könnte nun den Benutzer fragen :

"Handelt es sich um die Zahl 1; ja oder nein ?"

Da der Benutzer mit "nein" antwortet, könnte das Programm fragen :

"Handelt es sich um die Zahl 2; ja oder nein ?"

Wieder antwortet der Benutzer mit "Nein" und deshalb muß das Programm ziemlich lange fragen, bis es die Zahl, die der Benutzer sich gemerkt hat, gefunden hat.

Ganz abgesehen davon, daß es auf diese Weise kaum möglich ist, eine beliebige reelle Zahl zu finden (zum Beispiel 23.45307756), selbst im natürlichen Zahlenbereich dauert die Suche reichlich lange.

Deshalb ist beispielsweise die folgende Reihe von

Ja-Nein-Fragen sinnvoller, weil sie rascher zum Erfolg führen dürfte :

"Liegt die Zahl zwischen 0 und 10000;
ja oder nein ?"

Der Computerbenutzer müßte jetzt mit "ja" antworten und deshalb könnte das Programm als zweite Frage formulieren :

"Liegt die Zahl zwischen 0 und 5000;
ja oder nein ?"

Wieder ist mit "ja" zu antworten und die dritte Frage könnte deshalb lauten :

"Liegt die Zahl zwischen 0 und 2500;
ja oder nein ?"

Jetzt lautet die Antwort "nein" und deshalb weiß das "intelligente" Suchprogramm schon jetzt, daß die gesuchte Zahl zwischen 2500 und 5000 liegen muß. Also lautet sinnvollerweise die nächste Frage :

"Liegt die Zahl zwischen 2500 und 3750;
ja oder nein ?" usw.

Auf diese Weise wird das Suchprogramm durch "Eingrenzung" des in Frage kommenden Suchbereichs relativ rasch zum Sucherfolg gelangen. Ob dabei so vorgegangen werden muß, daß Schritt für Schritt der Suchbereich halbiert wird, oder ob man andere Suchalgorithmen verwendet, hängt von der jeweiligen Fragestellung ab - beispielsweise kann es sehr vorteilhaft sein, Zufalls - Suchbereiche zu bilden.

Auf Einzelheiten solcher Suchalgorithmen, die den Kern von praktisch nutzbaren Suchprogrammen ausmachen, also die Problemlösungen in diesem Anwendungsbereich ausmachen, wollen wir hier nicht eingehen; sie sind für die Zwecke dieses Buchs zu theoretisch. Details entnehme man deshalb der Spezialliteratur zu diesem Thema (siehe zum

Beispiel : HORN,W. : 1984 und die dort angegebene weiterführende Fachliteratur).

9.3 Das Beispiel

Das Beispiel, mit dessen Hilfe wir diesen speziellen Aspekt der künstlichen Intelligenz illustrieren wollen, stammt aus dem mathematischen Bereich :

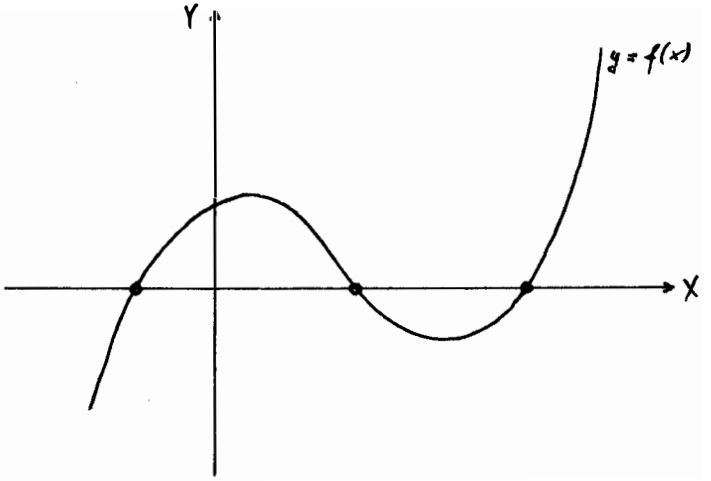
Es geht darum, die Nullstelle(n) einer zweidimensionalen mathematischen Funktion zu finden, ohne auf die Möglichkeiten algebraischer Lösungen zurückzugreifen oder darauf angewiesen zu sein.

Es liegt also eine analytisch beschriebene Funktion von der allgemeinen Form

$$y = f(x)$$

vor und es soll ein Computerprogramm entwickelt werden, welches diejenigen x-Werte bestimmt, an denen die jeweilige Funktion die waagrechte Achse schneidet (Nullstelle).

Zur Illustration einer solchen Funktion fügen wir die folgende Skizze hinzu :



Die Lösung des Suchproblems beschränkt sich in diesem einfachen Beispiel darauf, einen Weg zu finden, möglichst rasch die Positionen der Nullstelle(n) der Funktion aufzufinden. Den trivialen Weg, der darin besteht, einfach den Definitionsbereich der Funktion quasi von links nach rechts zu durchsuchen, wollen wir hier ausschließen. Den Suchpfad, den das Programm verfolgt, werden wir im folgenden Abschnitt zur Problemanalyse entwickeln.

9.4 Problemanalyse

Die Überlegungen zur Problemanalyse müssen an der Frage anknüpfen, welcher Suchpfad verfolgt werden soll, um die interessierenden Punkte der Funktion (vergl. obige Skizze) zu finden. Wir gehen dabei von dem folgenden Grundgedanken aus :

Innerhalb eines Intervalls (a,b) liegt mit Sicherheit eine Nullstelle, wenn die Vorzeichen von f(a) und f(b) verschieden sind (der Sonderfall, daß innerhalb dieses Intervalls auch mehrere Nullstellen liegen können, soll im folgenden Programm nicht berücksichtigt werden).

Ist nun das Vorzeichen von f((a+b)/2) gleich dem Vorzeichen von f(a), so liegt die Nullstelle im rechten Teilintervall (c,b), wobei c = (a+b)/2 und a < b.

In diesem Fall wird a gleich c gesetzt und das Verfahren wird wiederholt.

Entsprechend dieser Vorgehensweise wird b gleich c gesetzt, wenn die Nullstelle im linken Teilintervall liegt, das Vorzeichen von f(c) also gleich dem von f(b) ist.

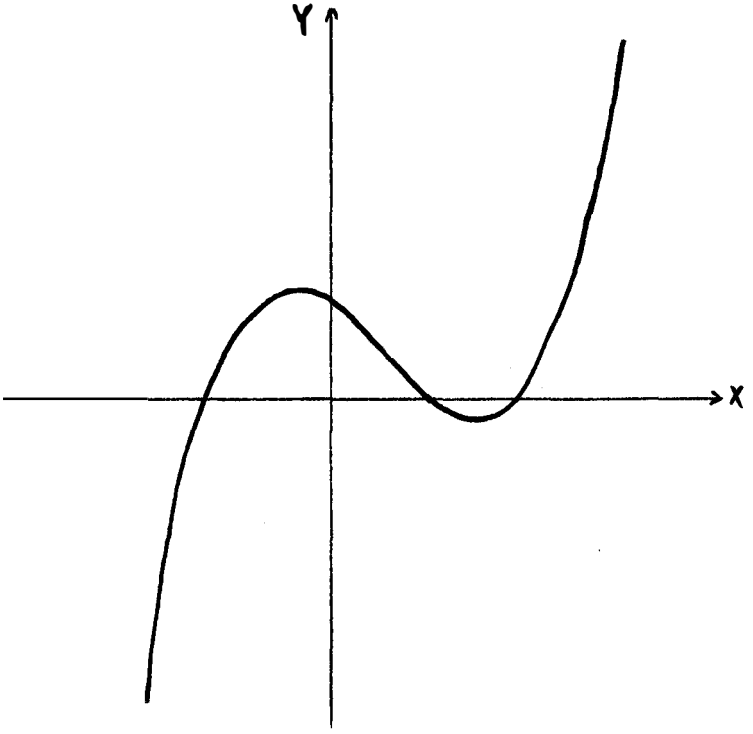
Dieses Verfahren läßt sich iterativ quasi unendlich oft wiederholen. Da dies aber nicht gewünscht sein kann, muß ein Abbruchkriterium vorgegeben werden. Dieses Abbruchkriterium kann in einem beliebig kleinen Wert bestehen, den die (ständig verkleinerte) Intervallgröße nicht unterschreiten soll; es könnte das jeweilige Abbruchkriterium aber auch an die Funktionswerte f(x) gekoppelt werden, in der Weise, daß der iterative Suchprozeß dann abgebrochen wird, wenn dieser Funktionswert nicht mehr als ein beliebig kleiner vorzugebender Wert von null abweicht. Wir haben im folgenden Programm den ersten Weg gewählt.

Weiterhin ist unabhängig von dieser gegebenen Genauigkeitsvorgabe im vorliegenden Programm die Maximalzahl der Intervallschachtelungen auf 500 begrenzt, um zu lange Rechenzeiten zu vermeiden, die entstehen würden, wenn immer weiter iteriert würde, um Funktionswerte noch näher am Wert null zu finden, obwohl dies vielleicht bei einer bestimmten Funktion gar nicht möglich ist.

Als Demonstrationsbeispiel wählen wir die folgende stetige Funktion :

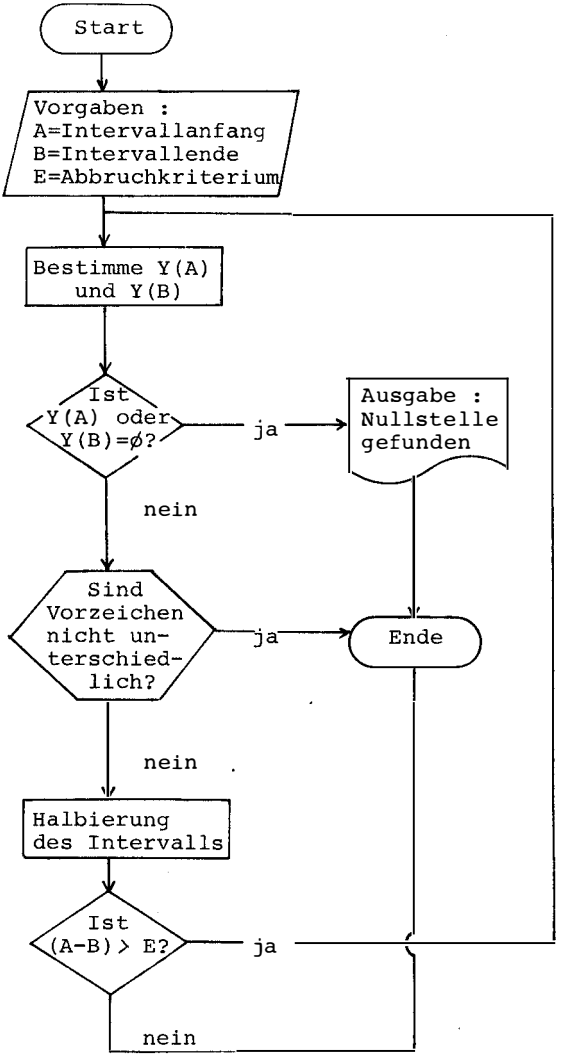
$$y = x^3/3 - (5/6)*x^2 - (3/2)*x + 3$$

Diese Funktion stellt sich graphisch folgendermaßen dar :



Zum Auffinden der Nullstelle dieser Funktion gemäß der oben vorgetragenen Überlegungen führt ein Programmablauf, der sich als Flußdiagramm folgendermaßen darstellt :

Flußdiagramm : Suchen



Dieses Flußdiagramm bedarf keiner näheren Erläuterungen.

Auf zwei "Spezialitäten" soll aber gesondert eingegangen werden :

1. Die auszugebenden Funktionswerte werden auf zwei Dezimalstellen gerundet;

2. Nach je fünf Iterationsrunden wird die Ausgabe auf dem Bildschirm unterbrochen, um das Hinauswandern der ersten Ergebnisse über den oberen Rand des Bildschirms zu verhindern.

Das Runden auf zwei Dezimalstellen macht die folgenden Arbeitsschritte erforderlich :

1. Multiplikation der zu rundenden Zahl mit 100 (wollen wir auf drei Stellen runden, so muß mit 1000 multipliziert werden etc.).

Aus der Zahl 12.343 wird 1234.3, aus 12.347 wird 1234.7

2. Hinzu-Addition von 0.5 :

Aus der ersten Zahl wird jetzt 1234.8, aus der zweiten wird 1235.2

Dadurch wird erreicht, daß alle Zahlen, die vor der Rundung auf 0 bis unter 5 enden, in der vorletzten Stelle sich nicht verändern, während alle Zahlen, die vor der Rundung auf 5 bis unter 0 enden, in der vorletzten Stelle um 1 erhöht werden.

3. Abschneiden der Dezimalstellen durch die INT-Funktion :

Aus der ersten Zahl wird jetzt 1234, aus der zweiten wird 1235.

4. Division durch 100 :

Jetzt wird der erste Schritt "rückgängig" gemacht. Deshalb wird aus der ersten Zahl 12.34, aus der zweiten wird 12.35, d.h. beide Zahlen sind korrekt auf zwei Dezimalstellen gerundet.

Diese vier Schritte lassen sich in folgender Anweisung zusammenfassen :

```
X  =  INT(X*100 + .5)/100
```

Das zweite Teilproblem, das hier gesondert beleuchtet werden soll, die Programmunterbrechnung nach jeder fünften Runde, ist sehr leicht zu programmieren :

Wenn ein Laufindex die Schachtelungsrunden "mitzählt" (im Programm wird dies durch die Variable I geleistet), dann ist zu unterbrechen, wenn I die Werte 5, 10, 15, 20, ... annimmt.

In all diesen Fällen gilt, daß I/5 = INT(I/5).

Also könnten wir programmieren :

```
IF I/5 = INT(I/5) THEN STOP
```

oder, was sich in dem folgenden Programm anbietet

```
IF I/5 = INT(I/5) THEN GOSUB 2000
```

9.5 Programm

```
10 REM KI03 - SUCHEN
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,     1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 3 :";
60 PRINT "      SUCHEN"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM DEMONSTRIERT EINEN PRO-"
90 PRINTTAB(10)"ZESS DES SUCHENS."
100 PRINT:PRINT:PRINT"ES SUCHT DIE NULSTELLEN EINER BELIEBI-"
110 PRINT"GEN STETIGEN FUNKTION INNERHALB EINES"
115 PRINT"       VORGEGEBENEN INTERVALLS."
118 GOSUB 2000:REM WARTEN
120 PRINTCHR$(147)
130 PRINT"DIE FUNKTION  F(X)  IST IN SATZ 350"
135 PRINTTAB(13)"VORGEGEBEN."
140 PRINT:PRINT  "SIE IST BELIEBIG AENDERBAR."
150 PRINT:PRINT:INPUT"INTERVALLANFANG A ";A
160 PRINT:  INPUT"INTERVALLENDE   B ";B
170 PRINT:PRINT"WIE KLEIN SOLL DAS INTERVALL"
180 INPUT "WERDEN ";E
185 PRINT:PRINTI;".SCHACHTELUNG :":PRINT
190 X=A:GOSUB 350:F1=Y
195 F1=INT(F1*100+.5)/100:PRINT"F(A) = ";F1;TAB(15)" ";
200 X=B:GOSUB 350:F2=Y
205 F2=INT(F2*100+.5)/100:PRINT"F(B) = ";F2
210 IF I/5=INT(I/5) THEN GOSUB 2000
220 IF F1=0 THEN PRINT"NULLSTELLE BEI X = ";A:GOTO 340
230 IF F2=0 THEN PRINT"NULLSTELLE BEI X = ";B:GOTO 340
240 IF F1<0 AND F2>0 OR F1>0 AND F2<0 THEN 270
250 PRINT"DIE VORAUSSETZUNG, VORZEICHEN F(A)<>"
260 PRINT"VORZEICHEN F(B), IST NICHT ERFUELLT.":GOTO 340
```

```
270 X=(A+B)/2:GOSUB 350
290 I=I+1:IF I>500 THEN I=I-1:GOTO 310
290 IF Y>0 AND F1>0 OR Y<0 AND F1<0 THEN A=X:F1=Y :GOTO 300
295 B=X:F2=Y
300 IF B-A>E THEN 185
305 PRINT:PRINT:PRINT"NULLSTELLE BEI : ":PRINT
310 PRINT"X     = ";X
320 PRINT"F(X) = ";Y
330 PRINT:PRINT"ANZAHL DER INTERVALLSCHACHTELUNGEN : ";I
340 PRINT:PRINT"ENDE DER BERECHNUNGEN":END
350 REM FUNKTION
360 Y = X↑3/3 - 5*X↑2/6 - 3*X/2 + 3
370 RETURN
2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(7)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
```

9.6 Variablenliste

A = Intervallanfang

B = Intervallende

E = Genauigkeitsvorgabe (Mindestintervall)

F1 = Funktionswert der linken Intervallgrenze

F2 = Funktionswert der rechten Intervallgrenze

I = Zählindex der Intervallschachtelungen

X = Funktionsvariable (X-Koordinate)

Y = Funktionswert f(x)

9.7 Programmbeschreibung

Satz 10-120 :

Überschrift, Erläuterungen, Löschen des Bildschirms

Satz 130-150 :

Weitere Erläuterungen zum Programm

Satz 150-180 :

Eingabe der Intervallgrenzen für den Start des Suchprozesses und der Genauigkeitsvorgabe für dessen Abbruch

Satz 185 :

Ausgabe der Laufnummer der Schachtelung, die gerade berechnet wird - beginnend bei 0

Satz 190-205 :

Durch einen jeweiligen Sprung in das Unterprogramm 350 (dort ist die Funktion, die überprüft werden soll, festgelegt) werden die Funktionswerte der Grenzen des jeweiligen Intervalls bestimmt und ausgegeben; vor der Ausgabe werden dann noch die Funktionswerte der besseren Lesbarkeit wegen auf zwei Dezimalstellen gerundet

Satz 210 :

Nach jeder fünften Schachtelungsrunde wird das Programm durch Sprung ins Unterprogramm 2000 zum Warten unterbrochen, damit der Leser in Ruhe die Ausgaben studieren kann, bevor die ersten Werte über den oberen Bildschirmrand hinaus verschwinden

Satz 220-230 :

Ist einer der Funktionswerte gleich null, so erfolgt eine entsprechende Meldung und es wird zum Programmende verzweigt

Satz 240 :

Sind die Funktionswerte der beiden Grenzen des Intervalls ungleich, dann kann die iterative Suchprozedur durch einen Sprung zum Satz 270 beginnen

Satz 250-260 :

Satz 250 wird nur erreicht, wenn die Vorzeichen der beiden Funktionswerte gleich sind; es wird dann ein entsprechender Kommentar ausgegeben und zum Programmende verzweigt

Satz 270 :

Das Intervall wird geteilt und der Funktionswert für c = (a+b)/2 wird durch einen Sprung ins Unterprogramm 350 berechnet

Satz 280 :

Der Index I, der die Intervallschachtelungen zählt, wird um 1 erhöht; wenn I nach der Erhöhung den Wert 501 erreicht hat (zweites Kriterium des Abbruchs), wird zur Ergebnisausgabe in Satz 310 verzweigt

Satz 290-295 :

Wenn das Vorzeichen des Funktionswertes f(c) (also der Intervallmitte) gleich dem Vorzeichen des Funktionswertes der linken Intervallgrenze ist, dann wird c zur (neuen) linken Intervallgrenze für den darauffolgenden Suchschritt; entsprechend wird gegebenenfalls rechts verfahren

Satz 300 :

Ist das neue Intervall größer als der vorgegebene Genauigkeitswert (erstes Abbruchkriterium), so erfolgt ein Rücksprung zum Satz 185, um den nächsten Suchschritt einzuleiten

Satz 305-330 :

Die näherungsweise bestimmte Nullstelle, der nahe beim Wert null liegende Funktionswert der Nullstelle und die Zahl der Schachtelungen, die zu diesem Ergebnis geführt haben, werden ausgegeben

Satz 340 :

Ende des Programms

Satz 350-370 :

Unterprogramm zur Berechnung und Ausgabe der Funktionswerte der jeweiligen Intervallgrenzen

Satz 2000-2040 :

Unterprogramm zum Warten

9.8 Programmergebnisse

Nach dem Start des Programms werden auf dem Bildschirm nach Überschrift und Erläuterungen die folgenden Informationen vom Benutzer angefordert :

INTERVALLANFANG A ?

INTERVALLENDE B ?

Geben wir für A zum Beispiel den Wert -5 und für B den Wert 5 ein, so verlangt das Programm weiterhin die folgende Information :

WIE KLEIN SOLL DAS INTERVALL
WERDEN ?

Antworten wir beispielsweise mit dem Wert 0.1, so ergibt sich :

0. SCHACHTELUNG :

F(A) = -52 F(B) = 16.33

1. SCHACHTELUNG :

F(A) = -52 F(B) = 3

2. SCHACHTELUNG :

F(A) = -3.67 F(B) = 3

3. SCHACHTELUNG :

F(A) = -3.67 F(B) = 2.92

4. SCHACHTELUNG :

```
F(A) = -3.67          F(B) = .69

 5. SCHACHTELUNG :

F(A) = -1.2           F(B) = .69

 6. SCHACHTELUNG :

F(A) = -.19           F(B) = .69


NULLSTELLE BEI :

X      =  -1.953125
F(X)   =     .267246241

ANZAHL DER INTERVALLSCHACHTELUNGEN  :  7

ENDE DER BERECHNUNGEN
```

9.9 Ausblick

Wir haben schon darauf hingewiesen, daß Suchprogramme im Grunde mit Bestandteile von Experten- und Auskunftssystemen sind. Dies soll auch hier nochmals betont werden.

Im Zusammenhang mit dem Beispiel dieses Kapitels muß daran erinnert werden, daß dieses Programm für beliebige stetige Funktionen eingesetzt werden kann. Es ist dazu lediglich erforderlich, eine entsprechende Änderung im Unterprogramm 350 vorzunehmen.

Wenn mit einer anderen Funktion gearbeitet wird, dann sollte man aber eine ungefähre, sehr grobe Vorstellung von der Lage einer Nullstelle haben, damit man die Grenzwerte A und B des ersten Intervalls sinnvoll vorgeben kann. Häufig ist zu diesem Zweck eine grobe Funktionsskizze hilfreich

oder ein vorgeschaltetes Graphikprogramm, welches die Funktion schnell einmal auf den Bildschirm zeichnet (siehe dazu etwa : VOSS,W. : 1984).

Schließlich ist anzumerken, daß bei Funktionen, die komplizierter sind, insbesonder bei mehrdimensionalen Funktionen der Suchaufwand beträchtlich werden kann. Es ist dann zu prüfen, ob es effizientere Suchalgorithmen, als das hier vorgestellte Verfahren gibt.

Kapitel 10 : Entscheiden

10.1 Aufgabenstellung

Im Bereich jeder empirischen Forschung , d.h. in all den Bereichen, die sich Erkenntnisse auf der Basis sinnlich wahrnehmbarer Erfahrungen verschaffen, muß man davon ausgehen, daß jede Aussage nichts anderes als eine Hypothese ist.

Beispielsweise ist die Aussage

"Die Einwohnerzahl der DDR beträgt 17 Millionen"

eine Hypothese - allein schon deshalb, weil sie empirischer Überprüfungen bedarf. Das gleiche gilt für die Aussage

"Das Durchschnittseinkommen in der Bundesrepublik liegt bei 2400 DM",

oder für die Aussage

"Es besteht ein Zusammenhang zwischen der Körpertemperatur eines Menschen und krankhaften Befunden".

An diesen und ähnlichen Beispielen erkennt man, daß praktisch jede Aussage - gleich welcher Art - als Untersuchungshypothese interpretiert werden kann.

Die Aufgabe der empirischen Wissenschaft besteht im Grunde darin, eine Entscheidung darüber herbeizuführen, ob eine derartige Hypothese im Lichte empirischer Befunde als bestätigt gelten kann, oder ob sie sinnvollerweise verworfen werden

muß.

Dabei gelten im Prinzip die in nachstehender Skizze verdeutlichten Zusammenhänge :

	Die Aussage ist in der Realität	
	wahr	falsch
Die Hypothese wird beibehalten	o.k.	II
verworfen	I	o.k.

In diesem Schema bedeutet "o.k." die Situation, in der eine korrekte Entscheidung getroffen wurde; "I" bedeutet, daß eine an sich wahre Hypothese (zu Unrecht) verworfen wurde (sog. Fehler vom Typ I); "II" bedeutet, daß eine an sich falsche Hypothese (zu Unrecht) beibehalten wurde (sog. Fehler vom Typ II).

Wir erkennen an diesem Schema, daß empirische Forschungen nie dazu taugen können, über Wahrheit oder Falschheit einer Aussage (Hypothese) zu entscheiden, sondern es geht nur darum, ob eine Hypothese beibehalten oder ob sie verworfen werden soll.

Gleichgültig, wie man entscheidet, es besteht immer die Gefahr, daß man falsch entschieden hat. Dies liegt letzten Endes daran, daß wir nie wissen können, in welcher Spalte des obigen Schemas wir uns befinden. Wir sind nicht im Besitz der absoluten Wahrheit - wenn das so wäre, bräuchten wir keine empirischen Forschungen und keine empirischen Wissenschaften mehr.

Was die empirische Wissenschaft lediglich leisten kann, ist, die Entscheidung über Beibehaltung oder Verwerfung von Hypothesen herbeizuführen und

darüberhinaus anzugeben - und das ist sehr wichtig - wie groß die Wahrscheinlichkeit einer Fehlentscheidung in diesem Zusammenhang ist.

Eine derartige Quantifizierung der jeweiligen Entscheidungsfehler ist eine der Hauptaufgaben der Wahrscheinlichkeitsstatistik und führt dann zu Aussagen etwa der folgenden Art :

"Die Hypothese, daß das Durchschnittseinkommen in der Bundesrepublik Deutschland 2400 DM beträgt, kann beibehalten werden, wobei das Fehlerrisiko (Wahrscheinlichkeit) 5 % beträgt"

oder positiv formuliert :

"Die Aussage, daß das Durchschnittseinkommen 2400 DM beträgt, ist mit 95 %iger Wahrscheinlichkeit richtig".

Der Leser kann sich sicherlich aufgrund dieser Ausführungen schon vorstellen, wie derartige Wahrscheinlichkeitsberechnungen in ein Expertensystem eingebaut werden könnten :

Hat man in einem medizinischen System auf der Grundlage eingegebener Symptome die Ergebnisse, daß die Krankheiten

- Lungenentzündung
- Bronchitis
- Bronchialkrebs

in Frage kommen könnten, so wäre dann nach entsprechenden Wahrscheinlichkeitsberechnungen diese Auskunft vielleicht folgendermaßen modifiziert :

- Lungenentzündung	72 %
- Bronchitis	87 %
- Bronchialkrebs	31 %

Dies bedeutet, daß die Aussage

"Der Patient leidet an einer Lungenentzündung"

mit einer Wahrscheinlichkeit von 72 % zutrifft und entsprechendes gilt auch für die anderen Diagnosen.

Wie man nun im Detail vorgehen muß, um über solche oder ähnliche Aussagen eine rationale Entscheidung herbeiführen zu können, beziehungsweise um nach der Entscheidung die Wahrscheinlichkeiten für eine Fehlentscheidung oder aber für eine korrekte Entscheidung berechnen zu können, wird im folgenden Abschnitt beschrieben.

10.2 Lösungsansätze

Die Vorgehensweise bei der Herbeiführung einer Entscheidung über eine vorher formulierte Hypothese und der folgenden Quantifizierung der Entscheidungswahrscheinlichkeiten - man spricht in diesem Zusammenhang von einem statistischen Test oder von einem Signifikanztest - ist leicht anhand eines einfachen Beispiels zu zeigen.

Die zu testende Hypothese möge folgendermaßen lauten :

"Der Anteilswert von Knabengeburten beträgt 0.5 (= 50 %)."

Um die Entscheidung darüber herbeizuführen, ob diese Hypothese beibehalten werden kann, oder ob sie zu verwerfen ist, benötigen wir empirische Befunde. Betrachten wir deshalb 10 zufällig ausgewählte Geburten (das Prinzip Zufall spielt in diesem Zusammenhang eine außerordentlich wichtige Rolle; d.h. wir müssen quasi per Losverfahren 10 Geburten auswählen) und notieren die Ergebnisse :

Knaben	7	(= K)
Mädchen	3	(= N-K)
Summe	10	(= N)

Wenn die Hypothese zutreffen würde, wären natürlich bei einem Umfang unserer Stichprobe von N = 10 genau K = 5 Knaben (und (N-K) = 5 Mädchen) zu erwarten gewesen. Da diese Erwartung sich nicht erfüllt hat, liegt der Gedanke nahe, die zu prüfende Hypothese aufgrund des empirischen Befundes zu verwerfen.

Ist dies aber wirklich sinnvoll ?

Wir müssen immerhin bedenken, daß wir (nur) 10 zufällige Beobachtungen zur Grundlage unserer Entscheidung benutzen, d.h. unser Ergebnis (7 Knaben und 3 Mädchen) kann ja "reiner Zufall" sein und die Hypothese trifft in Wahrheit doch zu.

Deshalb stellen wir jetzt die folgende sehr wichtige Überlegung an :

Wenn die Wahrscheinlichkeit dafür, daß eine Stichprobe wie die von uns beobachtete (7 K und 3 M) oder eine noch weiter von der Behauptung der Hypothese abweichende Stichprobe (8 K und 2 M, 9 K und 1 M, 10 K und 0 M) sehr klein wird, unter der Bedingung, daß die Hypothese in Wahrheit zutrifft, dann werden wir die Hypothese verwerfen.

Anders ausgedrückt : Wenn etwas beobachtet wird, was bei Gültigkeit der Hypothese eigentlich garnicht auftreten dürfte (eben nur mit einer sehr geringen Realisationswahrscheinlichkeit behaftet ist), dann zweifeln wir eben an dieser Gültigkeit, d.h. wir verwerfen die Hypothese.

Ist die Realisationswahrscheinlichkeit des Stichprobenbefundes (oder eines noch weiter abweichenden Befundes) hingegen bei Unterstellung der Gültigkeit der Hypothese hoch, wenn also etwas

passiert, was eigentlich auch zu erwarten war, dann behalten wir die Hypothese bei, wir betrachten sie als bestätigt.

Was bedeutet in diesem Zusammenhang "geringe Realisationswahrscheinlichkeit" ?

In der Regel gibt man sich einen kleinen Prozentwert vor (zum Beispiel 5 %), der in der Teststatistik " Signifikanzniveau " genannt wird. Wir wollen diesen Prozentsatz im folgenden mit dem Symbol A und die zu berechnende Wahrscheinlichkeit (der Realisation der Stichprobe oder eines noch weiter von der Hypothese abweichenden Befundes) mit der Abkürzung UW. Mit W schließlich kürzen wir einfach das Wort Wahrscheinlichkeit ab.

Mit diesen Symbolen können wir die obigen Überlegungen sehr viel kürzer fassen, wenn wir zusätzlich die Knabenanzahl in Zufallsstichproben vom Umfang N = 10 einfach mit K bezeichnen :

Wenn W(K>=7), unter der Bedingung, daß die Hypothese (= HO) zutrifft, kleiner oder gleich A ist, dann verwerfen wir die Hypothese, andernfalls gilt sie als bestätigt.

Noch weiter abgekürzt, sieht dies folgendermaßen aus :

Ist UW(K>=7)|HO <= A , dann Verwerfung

Entscheidend ist also, die Wahrscheinlichkeiten dafür auszurechen, daß in einer Zufallsstichprobe vom Umfang 10

- 7 Knabengeburten oder
- 8 Knabengeburten oder
- 9 Knabengeburten oder
- 10 Knabengeburten

auftreten, unter der Bedingung, daß die zu prüfende Hypothese zutrifft, daß also die

Wahrscheinlichkeit für eine Knabengeburt 0.5 beträgt.

Bezeichnet man diese Wahrscheinlichkeit des einzelnen Ereignisses mit PI, so ergibt sich die Wahrscheinlichkeit dafür, daß bei N=10 Geburten genau K=7 Knabengeburten auftreten, gemäß der folgenden Formel :

$$W(K) = \binom{N}{K} * PI^{K} * (1-PI)^{N-K}$$

Dabei bedeutet :

$$\binom{N}{K} = N!/(K! * (N-K)!)$$

und

$$N! = N * (N-1) * (N-2) * \ldots * 3 * 2 * 1$$

(lies : N - Fakultät)

Zur Herleitung der obigen Berechnungsformel für W(K), der sog. Binomialverteilung , siehe z.B. TIEDE,M./VOSS,W. : 1982.

In unserem Beispiel ergibt sich :

$$W(K=7) = \binom{10}{7} * PI^{7} * (1-PI)^{3} =$$

$$= 120 * 0.5^{7} * 0.5^{3} = 0.117\ldots$$

Die Wahrscheinlichkeit dafür, daß bei 10 Geburten sieben Knaben- und drei Mädchengeburten beobachtet werden, wenn die Wahrscheinlichkeit für eine Kabengeburt bei einem "Versuch" 0.5 beträgt (wenn also die Gültigkeit der zu testenden Hypothese

unterstellt wird), beträgt ca. 11.7 %.

Entsprechend erhalten wir für acht Knaben- und zwei Mädchengeburten :

$$W(K=8) = \binom{10}{8} * 0.5^{8} * 0.5^{2}$$

$$= 45 * 0.5^{8} * 0.5^{2} = 0.044...$$

Für neun Knaben- und eine Mädchengeburt erhalten wir :

$$W(K=9) = \binom{10}{9} * 0.5^{9} * 0.5^{1}$$

$$= 10 * 0.5^{9} * 0.5^{1} = 0.0098...$$

und schließlich für zehn Kaben- und null Mädchengeburten :

$$W(K=10) = \binom{10}{0} * 0.5^{10} * 0.5^{0}$$

$$= 1 * 0.5^{10} * 0.5^{0} = 0.00098...$$

Die Wahrscheinlichkeit schließlich, in einer Zufallsstichprobe vom Umfang N = 10, sieben oder mehr Knabengeburten (und entsprechend drei oder weniger Mädchengeburten) zu beobachten, ergibt sich als die Summe der vier berechneten Wahrscheinlichkeiten, also :

$$UW(K>=7 | PI=0.5) =$$

$$W(7) + W(8) + W(9) + W(10) =$$

$$0.117 + 0.044 + 0.0098 + 0.00098 =$$

$$0.17178$$

Die Wahrscheinlichkeit, nach der wir suchen - man nennt sie Überschreitungswahrscheinlichkeit - beträgt also 17.178 %.

Wenn wir uns als Signifikanzniveau A z.B. 5 % vorgegeben haben, so bedeutet dieses Ergebnis, daß eine empirische Beobachtung gemacht wurde, deren Wahrscheinlichkeit nicht gering genug war, um die Hypothese zu verwerfen.

Da die Überschreitungswahrscheinlichkeit größer ist als das Signifikanzniveau gilt die zu prüfende Hypothese als bestätigt.

Glücklicherweise kann diese umständliche Art der Berechnung, die allerdings für einen Computer nicht sonderlich problematisch ist, wenn er über das entsprechende Programm verfügt, deutlich abgekürzt werden, wie wir im folgenden Abschnitt zeigen.

10.3 Das Beispiel

Im Demonstrationsbeispiel zu diesem Kapitel wollen wir beim Muster des statistischen Tests eines Anteilswerts der Einfachheit halber bleiben. Die zu testende Hypothese laute folgendermaßen :

"Der Anteil von Arbeitern an den abhängig Beschäftigten in der Bundesrepublik Deutschland beträgt 0.48 (= 48 %)".

Diese Hypothese soll mit einem Signifikanzniveau von A = 10 % getestet werden.

Zur Gewinnung der empirischen Datenbasis ziehen wir eine Zufallsstichprobe vom Umfang N = 400 und stellen in dieser Stichprobe einen Arbeiteranteil von 45 % fest.

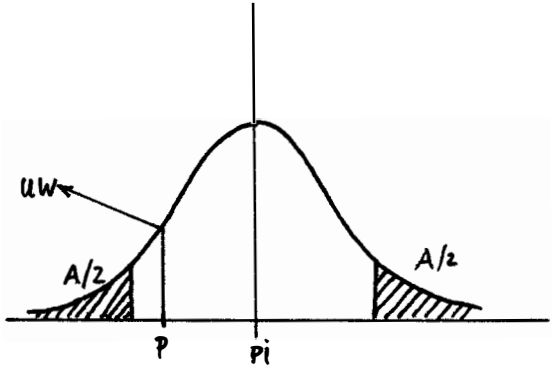
Gemäß den Überlegungen des vorangegangenen

Abschnitts müssen wir nun die Wahrscheinlichkeit UW (= Überschreitungswahrscheinlichkeit) dafür berechnen, daß - Gültigkeit der zu prüfenden Hypothese unterstellt - ein solcher oder ein noch weiter von der Hypothese abweichender Stichprobenbefund realisiert wird.

Dies ist mit der Binomialverteilung des vorangegangenen Abschnitts ein sehr umständliches Unterfangen, weil wir die Wahrscheinlichkeiten W(180) (= 45 % von 400), W(179), W(178), ... W(2), W(1) und W(0) berechnen müßten, insgesamt also 181 Einzelwahrscheinlichkeiten - auch dies ist allerdings per Computer kein allzu großes Problem.

Einfacher geht es, wenn man weiß, daß bei großen Stichproben (Faustregel : N $>$30), die Binomialverteilung mit hinreichender Güte durch die Gauß-Verteilung, die sog. Normalverteilung ersetzt werden kann.

In unserem Beispiel sieht sie so aus, wie die folgende Skizze, und die jetzt noch gesuchte Überschreitungswahrscheinlichkeit ist nichts anderes als die Fläche unter dieser Kurve links vom Wert P (P bezeichnet den realisierten Anteilswert in der Stichpobe).



Wenn diese Überschreitungswahrscheinlichkeit größer ist als die schraffierte Fläche A/2, die den linken Bereich des Signifikanzniveaus A darstellt (wir mußten in diesem Fall das Signifikanzniveau zweiseitig anlegen, weil vor der Stichprobenziehung ja nicht bekannt ist, ob der realisierte Anteilswert P größer oder kleiner als der Hypothesenwert sein wird), wenn also P nicht im schraffierten Bereich liegt, kann die Hypothese als bestätigt gelten, andernfalls wird sie verworfen.

Für den Nicht-Statistiker sind diese Ausführungen und auch die des vorangegangenen Abschnitts sicherlich nicht sehr leicht zu verstehen. Wir verweisen deshalb für den interessierten Leser auf die einführende wahrscheinlichkeitsstatistische Literatur, zum Beispiel TIEDE,M. / VOSS,W. : 1982).

10.4 Problemanalyse

Soll ein Anteilswerttest per BASIC-Programm durchgeführt werden, so müssen zunächst als Input-Informationen der in der zu testenden Hypothese behauptete Anteilswert PI, sowie das Signifikanzniveau A und der Stichprobenanteilswert P vorgegeben werden.

Wenn die beiden Anteilswerte, also die Werte PI und P, übereinstimmen, kann der Test mit der Beibehaltung der Hypothese schon abgebrochen werden.

Ist dies nicht der Fall, ist die Größe des Stichprobenumfangs, der Wert N einzugeben und in Abhängigkeit davon zu entscheiden, ob per Binomial- oder per Normalverteilung getestet wird.

Der Test selbst besteht einfach nur in der Berechnung der Überschreitungswahrscheinlichkeit UW, wie oben beschrieben, und deren Vergleich mit dem Signifikanzniveau A. Ist UW > A/2, bleibt die Hypothese beibehalten, andernfalls wird sie verworfen.

Beim Test mit der Binomialverteilung ist auf folgende Besonderheiten zu achten :

Im Rahmen der Berechnung der Binomialkoeffizienten (vergl. Abschnitt 10.2) wird mit Fakultäten gearbeitet.

Dabei gelten die folgenden mathematischen Vereinbarungen :

$$0! = 1 \qquad \text{und} \qquad \binom{N}{N} = 1$$

Die Berechnung dieser Binomialkoeffizienten kann wesentlich durch Kürzen vereinfacht werden. Um diese Möglichkeit optimal nutzen zu können,

vertauschen wir K mit (N-K), wenn K > N/2. Dieser Tausch erleichtert das Kürzen und ist erlaubt, weil die folgende Beziehung gilt :

$$\binom{N}{K} = \binom{N}{N-K}$$

Auch zum Test mit der Normalverteilung noch eine Anmerkung : Die Bestimmung der Größe UW (Überschreitungswahrscheinlichkeit) kann so erfolgen (vergl. auch Skizze in Abschnitt 10.3), daß wir in kleinen Intervallen vom Punkt P aus nach links wandern und die Rechtecksflächen über diesen Intervallen aufsummieren.

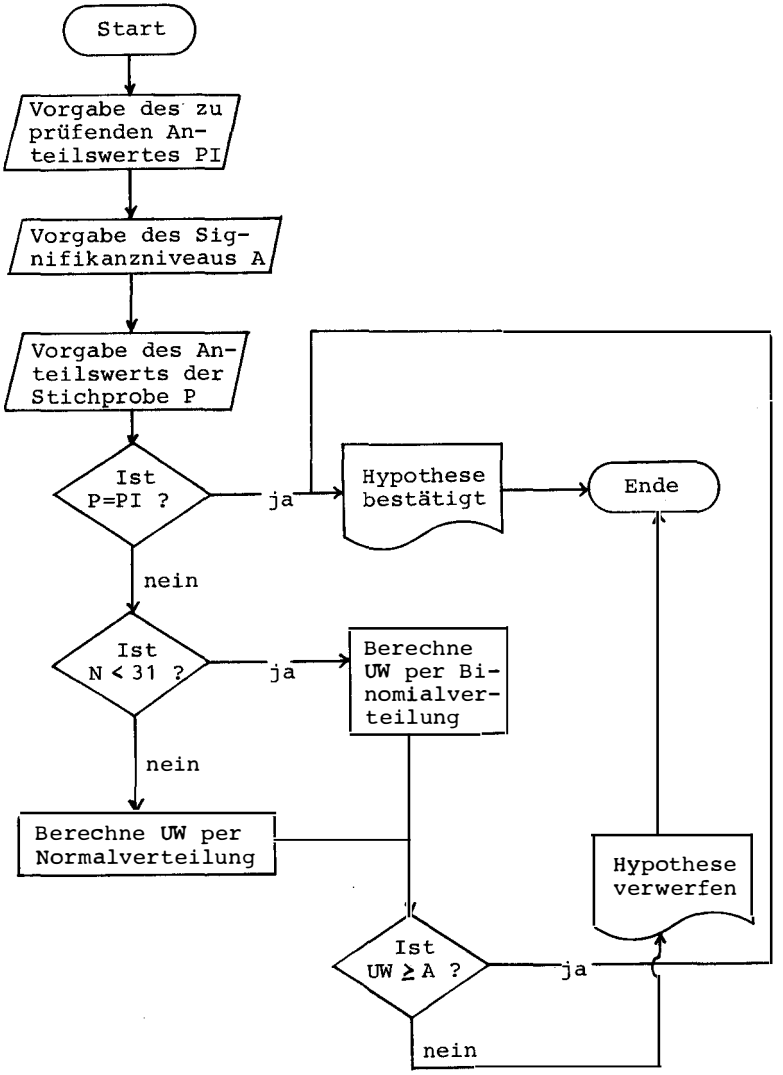
Zu diesem Zweck ist es nur erforderlich, die Ordinaten der Dichtefunktion der Normalverteilung in den jeweiligen Abszissenpunkten mit der Intervallbreite zu multiplizieren, was die schwierige Integration der Kurve entbehrlich macht.

Die Dichtefunktion dieser Kurve hat die folgende Form :

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Somit gelangen wir zu dem folgenden generellen Programmablaufplan :

Flußdiagramm Entscheiden (Anteilswerttest)



10.5 Programm

```
10 REM KI04 - ENTSCHEIDEN
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,     1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 4 :";
60 PRINT "ENTSCHEIDEN"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM SIMULIERT ENTSCHEIDUNGEN":PRINT
90 PRINT"ES ENTSCHEIDET DARUEBER, OB EINE HYPO-"
100 PRINT"THESE UEBER EINEN ANTEILSWERT IM LICHTE"
110 PRINT"EMPIRISCHER BEFUNDE BEIBEHALTEN WERDEN"
120 PRINT"SOLLTE, ODER OB SIE ZU VERWERFEN IST."
130 PRINT:PRINT"(SOG. PARAMETRISCHER HYPOTHESENTEST "
140 PRINTTAB(8)"ODER SIGNIFIKANZTEST)."
150 PRINT:PRINT:PRINT:PRINT
160 PRINT"DIESES PROGRAMM BENOETIGT ALS INPUT-"
170 PRINT"INFORMATIONEN : ":PRINT
180 INPUT "BEHAUPTETER ANTEILSWERT IN % : ";PI
190 INPUT "SIGNIFIKANZNIVEAU       IN % : ";A
200 INPUT "STICHPROBENUMFANG            : ";N
210 PRINT "ANTEILSWERT IN DER"
220 INPUT "   STICHPROBE            IN % : ";P
225 A$=" IST NICHT WIDERLEGT."
230 IF P=PI THEN UW=50:GOTO 600:REM AUSGABE
240 IF N<31 THEN 370:REM BINOMIAL
250 REM NORMAL
260 S=SQR((PI*(100-PI))/N)
270 KA=ABS((P-PI)/S)
```

```
280 REM UEBERSCHREITUNGSWAHRSCHEINLICHKEIT
290 PRINTCHR$(147):PRINTTAB(7)"MOMENT BITTE, ICH RECHNE":
    PRINT:PRINT
300 FOR I=KA-0.005 TO 4 STEP 0.01
310 F=(1/SQR(2*π))*EXP(-I*I/2)
320 W=F*0.01:UW=UW+W
330 NEXT I
340 UW=UW*100:UW=INT(UW*100+.5)/100
350 IF UW <=A/2  THEN A$=" IST WIDERLEGT."
360 GOTO 600
370 REM BINOMIAL
380 K=P*N/100:K=INT(K+.5):PI=PI/100
390 Y=N-K:P=P/100:Q=1-P
395 KA=K
400 SA=KA:SE=0:SW=-1
410 IF K>N/2 THEN SE=N:SW=1
415 FOR K=SA TO SE STEP SW
416 IFK=0 OR K=N THEN BK=1:GOTO 490
420 P1=K:IF K=0 OR K=1 THEN P1=1:GOTO440
430 FOR I=K-1 TO 1 STEP-1:P1=P1*I:NEXTI
440 PK=P1:P1=N
450 IF PK=1 THEN PN=N:GOTO 480
460 FOR I=N-1 TO N-K+1 STEP-1:P1=P1*I:NEXT I
470 PN=P1
480 BK=PN/PK
490 W=BK*PI↑K*(1-PI)↑(N-K)
510 UW=UW+W
520 NEXT K
530 UW=UW*100:UW=INT(UW*100+.5)/100
535 PI=PI*100
540 IF UW<=A/2 THEN A$=" IST WIDERLEGT.":GOTO 600
600 PRINTCHR$(147)
610 PRINTTAB(15)"ERGEBNIS :":PRINT:PRINT
620 PRINT"DIE HYPOTHESE, DASS DER ANTEILSLSWERT"
630 PRINTPI;" % BETRAGE,";A$
640 PRINT:PRINT"(DIE UEBERSCHREITUNGSWAHRSCHEINLICHKEIT"
650 PRINT"BETRAEGT ";UW;" %)"
998 PRINT:PRINT:PRINT"ENDE DER AUSGABE"
999 END
2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
```

10.6 Variablenliste

A	=	Signifikanzniveau
A$	=	Stringvariable für Tastatureingabe bzw. Hilfsstring für die Ausgabe
BK	=	Binomialkoeffizient
F	=	Dichte der Normalverteilung (Ordinate)
I	=	Laufindex
J	=	Laufindex
K	=	Werte der Stichprobenvariablen (mögliche Sichprobenbefunde)
KA	=	Stichprobenbefund als Absolutwert
N	=	Stichprobenumfang
P	=	Stichprobenanteilswert
PI	=	Grundgesamtheitsanteilswert (Hypothesenwert)
PK	=	K!
PN	=	N!
P1	=	Produkt Nr.1 in der Berechnung des Binomialkoeffizienten
Q	=	1-P (100 - P in %)
S	=	Streuung der Normalverteilung
SA	=	Schleifenanfang in der Bestimmung der Fakultäten in BK
SE	=	Schleifenende

SW = Schrittweite in dieser Schleife

UW = Überschreitungswahrscheinlichkeit

W = Einzelwahrscheinlichkeit

Y = N-K

10.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Warten

Satz 80-150 :

Erläuterungen zum Programm

Satz 160-220 :

Anforderung der Input-Informationen

Satz 225 :

Belegung eines Hilfsstrings für die Ergebnisausgabe

Satz 230 :

Testentscheidung, wenn der Stichprobenbefund mit dem behaupteten Wert übereinstimmt; Sprung zum Satz 600 zur Ergebnisausgabe

Satz 240 :

Verzweigung zum Test mit der Binomialverteilung

(Satz 370 ff.), wenn der Sichprobenumfang zu klein ist

Satz 250-270 :

Beginn des Tests mit der Normalverteilung (bei Stichprobenumfängen, die größer als 30 sind); Bestimmung der Streuung der Verteilung (260) und Bestimmung des standardisierten Stichprobenbefunds (270)

Satz 280 :

Beginn der Bestimmung der Größe UW (Überschreitungswahrscheinlichkeit)

Satz 290 :

Löschen des Bildschirms und Ausgabe einer Wartemeldung, weil die Berechnung etwas Zeit in Anspruch nimmt

Satz 300-330 :

Berechnung der einzelnen Wahrscheinlichkeiten als Rechtecksflächen (siehe Abschnitt 10.4) und Aufsummierung des Wertes UW

Satz 340 :

Runden der Überschreitungswahrscheinlichkeit

Satz 350 :

Herbeiführung der Testentscheidung durch Vergleich der Überschreitungswahrscheinlichkeit mit dem Signifikanzniveau

Satz 360 :

Sprung zum Satz 600 zur Ergebnisausgabe

Satz 370 :

Beginn des Tests mit der Binomialverteilung bei

kleinem Stichprobenumfang (Satz 370 wird nur erreicht, wenn N < 31)

Satz 380-390 :

Festlegung der Ausgangswerte : Absolutbetrag des Stichprobenbefundes (K), für das einzelne Ereignis behauptete Eintrittswahrscheinlichkeit (PI), Rest-Absolutbetrag (Y=N-K), Stichprobenanteilswert (P) und Komplementärwert (Q=1-P)

Satz 395 :

Notierung des K-Wertes im Feld KA, um diesen Wert für die Formulierung der Ergebnisausgabe noch zur Verfügung zu haben, auch wenn K mit N-K vertauscht wurde, weil K > N/2 (siehe Satz 410)

Satz 400 :

Belegung der Schleifenparameter (Anfangswert, Endwert und Schrittweite) für die Berechnungen der Fakultäten, die in der Bestimmung des Binomialkoeffizienten benötigt werden

Satz 410 :

Wenn K > N/2, dann läuft die Schleife nicht von K bis 0 mit der Schrittweite -1, sondern von K bis N mit der Schrittweite N (vergl. dazu auch die Anmerkung zur vereinfachten Berechnung des Binomialkoeffizienten in Abschnitt 10.4)

Satz 415-520 :

Berechnung der Überschreitungswahrscheinlichkeit UW per Binomialverteilung;

dazu ist zunächst der Binomialkoeffizient zu berechnen, der den Wert 1 hat, wenn K = 0 oder wenn K = N (416 mit Sprung nach 490);

als nächstes ist K! zu bestimmen. Dies liegt mit 1 fest, wenn K = 0 oder K = 1 ist (420 mit Sprung nach 440);

wenn K ≠ 0 oder K ≠ 1, dann wird K! in einer Produktschleife berechnet (430) und das Ergebnis dieser Schleife wird unter PK gespeichert (440); zur Vorbereitung der nächsten Fakultätenberechnung (N!) wird P1 mit N belegt (440);

wenn sich ergeben hat, daß PK (= K!) = 1, dann muß gelten : PN (= N!) = N (450 mit Sprung nach 480); ist dies nicht der Fall, muß auch N! ausgerechnet werden, was in Satz 460 geschieht; das Ergebnis wird in PN gespeichert (470);

nach diesen Vorbereitungen kann in Satz 480 der Binomialkoeffizient BK ausgerechnet werden und in Satz 490 kann dann die Wahrscheinlichkeit W gemäß der Binomialverteilung bestimmt werden;

in Satz 510 wird diese Wahrscheinlichkeit von Schleifendurchlauf zu Schleifendurchlauf zur Überschreitungswahrscheinlichkeit UW aufkumuliert

In Satz 520 wird diese Schleife beendet

Satz 530-540 :

Runden der interessierenden Werte und Herbeiführung der Testentscheidung

Satz 600-650 :

Ausgabe der jeweiligen Testentscheidung

Satz 998-999 :

Beendigung des Programms

Satz 2000-2040 :

Unterprogramm zum Abwarten einer Tastatureingabe

10.8 Programmergebnisse

Wenn wir dieses Programm starten, so werden nach Überschrift und Erläuterungen die Input-Informationen angefordert, für die wir beispielsweise die folgenden Werte eingeben :

```
BEHAUPTETER ANTEILSWERT IN %      :  40
SIGNIFIKANZNIVEAU       IN %      :  10
STICHPROBENUMFANG                 :  100
ANTEILSWERT IN DER
   STICHPROBE           IN %      :  45
```

Nach dieser Eingabe benötigt das Programm etwas Rechenzeit und meldet dann :

```
                ERGEBNIS :

DIE HYPOTHESE, DASS DER ANTEILSWERT
 40 % BETRAGE, IST NICHT WIDERLEGT.

(DIE UEBERSCHREITUNGSWAHRSCHEINLICHKEIT
BETRAEGT  15.61  %)

ENDE DER AUSGABE
```

10.9 Ausblick

Ergänzend ist darauf hinzuweisen, daß wir noch nicht gezeigt haben, wie die Wahrscheinlichkeiten von Fehlentscheidungen (vergl. auch Abschnitt 10.1) berechnet werden können. Für derartige Aufgaben sollte die zuständige Fachliteratur zu Rate gezogen werden (siehe zum Beispiel : TIEDE,M./VOSS,W. : 1982, Band 1).

Weiterhin ist zu erwähnen, daß derartige Tests nicht nur auf Anteilswerte beschränkt sind. Sie können mit im Prinzip dem gleichen Instrumentarium auch für Durchschnittswerte, für Streuungen, für Zusammenhangsmaße etc. durchgeführt werden, wobei nur jeweils die zugrunde zu legende Wahrscheinlichkeitsverteilung zu modifizieren ist.

Schließlich soll auch auf die große Zahl spezieller Testverfahren verwiesen werden, die in den statistischen Lehrbüchern unter der Überschrift "nichtparametrische Testverfahren" vorgestellt werden (siehe dazu etwa : TIEDE,M./VOSS,W. : 1982, Band 2).

Kapitel 11 : Erkennen

11.1 Aufgabenstellung

In der Literatur zur künstlichen Intelligenz spielt das Stichwort "Mustererkennung" eine ganz zentrale Rolle (siehe z.B. BUCHBERGER,E. : 1984,I).

Wir müssen dabei von vornherein berücksichtigen, daß "Erkennen" im Sinne des Einsatzes von Computern unter ganz verschiedenen Gesichtspunkten betrachtet werden kann :

"Erkennen" kann zum Beispiel das Erfassen und sachgerechte Interpretieren menschlicher Sprache bedeuten; es kann aber auch etwa das optische Erkennen bestimmter Sachverhalte bedeuten und eventuell das Auslösen angemessener Reaktionen auf das "Erkannte" u.v.a.

Selbst das Erkennen von Schriften, d.h. von Buchstaben, Sonderzeichen und Ziffern durch geeignete Lesegeräte kann mit in diesen Bereich des Computereinsatzes aufgenommen werden.

Diese und viele andere Beispiele werden häufig unter dem Etikett " Mustererkennung " zusammengefaßt, wobei man sich unter diesem Stichwort in erster Stelle das optische Erkennen vorstellt. Man denke etwa an die automatisierte Auswertung von Satellitenphotos, von Röntgenaufnahmen, von Fernsehaufzeichnungen bei den verschiedenen Systemen der Verkehrsüberwachung u.ä.

Nicht zu vergessen sind in diesem Zusammenhang die vielfältigen Probleme, die bei der Steuerung von

Robotern gelöst werden müssen, wobei diese Lösungen unter anderem natürlich "Sehen" erfordern.

Es versteht sich fast von selbst, daß die Entwicklung von Programmen, die einen Computer "erkennen" lassen können, etwa in dem Sinne, daß er Bilder erkennt, gesprochene Worte versteht oder Handschriften lesen kann, außerordentlich schwierig ist. Allerdings hat man schon einzelne sehr bemerkenswerte Erfolge erzielen können :

Es gibt beispielsweise Großrechner, die in der Lage sind - dank ihrer Speicherkapazität und ihrer Rechengeschwindigkeiten - von Kameras aufgenommene Bilder zu interpretieren; bei einfachen graphischen Mustern gelingt dies heute selbst kleineren Rechnern (ein anschauliches Beispiel findet sich bei HORN,W. : 1984, Seite 41 ff.).

Rechner, die genormte Schriften über dafür geeignete Lesegeräte aufnehmen können, gibt es schon seit mehreren Jahrzehnten und Rechner, die Handschriften lesen können, lassen nicht mehr lange auf sich warten (die Informationseingabe über ein sog. Tablett , mit dem beliebig gekrümmte Linien "in den Rechner gechrieben" werden können, ist ja vielleicht ein erster wichtiger Schritt in diese Richtung).

Und schließlich ist man auch dabei, den rein akustischen Informationsaustausch zu ermöglichen : Schon seit einigen Jahrenn gibt es Programme, die geschriebene Texte in akustische Ausgaben umformen und über den Lautsprecher hören lassen können und seit kurzem besteht auch die Möglichkeit, über ein Mikrophon dem Rechner Informationen akustisch zukommen zu lassen. Letzten Endes gehört auch der sich derzeit rasch verbreitende Austausch von Informationen über die sog. Modems in diesen Bereich.

Wie kompliziert die Probleme des Erkennens im einzelnen sind, kann sehr anschaulich an der

Aufgabenstellung der Bilderkennung demonstriert werden : Unterschiedliche Beleuchtungen, unterschiedliche Blickwinkel, unterschiedliche Entfernungen können ein und dasselbe Objekt so stark verändern, daß die Leistungsfähigkeit des erkennenden Computerprogramms rasch an Grenzen stößt.

Das menschliche Gehirn ist auf der Grundlage früherer Erfahrungen in der Lage, die notwendigen Verrechnungen, die erkennen lassen, daß es sich immer noch um das gleiche Objekt handelt, so rasch zu vollziehen, daß sie uns kaum bewußt werden - ein Rechner hat da sehr viel größere Probleme.

Wir verweisen auf die sehr illustrativen Ausführungen bei STEDE,M. u.a. : 1984, S. 106 - 122.

11.2 Lösungsansätze

Ganz prinzipiell gibt es zwei Wege, wie Erkennen - sei es das akustische, das optische oder eine andere Art des Erkennens - vor sich gehen kann :

1. Ganzheitliche Methode
2. Analytische Methode

Im ersten Fall wird versucht, den zu erkennenden Sachverhalt als Ganzes aufzunehmen und zu interpretieren. Ein geschriebenes Wort zum Beispiel wird also als Ganzes und nicht durch Zusammensetzen von einzelnen Symbolen (Buchstaben) aufgenommen.

Soll diese ganzheitliche Informationsaufnahme gelingen, muß saubere Artikulation (bei gesprochenen Worten), saubere und unverwechselbare Schrift (bei geschriebenen Worten) bzw. scharfe und eindeutige Konturen (bei Bildern und Mustern) vorausgesetzt werden.

Weitere Voraussetzung ist ein großer Speicher des Rechners, weil ja Erkennen letzten Endes nur so funktionieren kann, daß er die aufgenommenen Informationen mit denjenigen Informationen vergleicht, die ihm schon zur Verfügung stehen - je mehr er davon hat, desto eher wird er in der Lage sein, Input-Informationen nach der Aufnahme auch zu erkennen.

Bei der analytischen Methode des Erkennens wird ein anderer Weg beschritten : Hier geht es darum, beispielsweise ein Wort Buchstabe für Buchstabe zu erkennen, ein akustisches Signal durch Zusammensetzen unter Umständen sehr vieler unterschiedlicher Tonschwingungen, ein Bild durch Zerlegen in möglichst kleine, einzelne Bildeinheiten.

Es leuchtet ein, daß auch für diese Vorgehensweise große Speicher benötigt werden, und daß die Rechenzeiten um so größer werden, je sorgfältiger bei der Analyse vorgegangen wird, d.h. je feiner in Einzelinformationen aufgeteilt wird. Selbst moderne Großrechner sind hier rasch überfordert :

"... der Rechner muß bei höherer Auflösung alle hundertstel Sekunde ungefähr hunderttausend Bildpunkte ablesen (es geht um das Erkennen bewegter Bilder, beispielsweise bei der Verkehrsüberwachung, der Verf.), ihnen einen Grauwert zuweisen UND das Ganze im logischen Teil analysieren. Da kommt man schnell auf zwanzig, dreißig Millionen Rechenoperationen in der Sekunde..." (HORX,M. : 1984, S.134).

Ein kleiner Homecomputer, wie der Commodore C64 ist sicherlich nicht in der Lage, in diesem Bereich der künstlichen Intelligenz einen aufsehenerregenden Beitrag zu leisten. Gleichwohl aber können einige wichtige Grundprinzipien von Programmen, die "Erkennen" leisten sollen, auch an diesem Gerät demonstriert werden, wie das folgende Beispiel zeigt.

11.3 Das Beispiel

Wir haben uns für dieses Kapitel wieder ein Beispiel ausgedacht, das ausschließlich mit der Erstausstattung des C 64-Benutzers bewältigt werden kann, also mit dem Rechner selbst und dem angeschlossenen Fernseher oder Monitor.

Echte Programme unter dem Etikett "Erkennen" oder "Mustererkennung" brauchen natürlich periphere Geräte : Große externe Speicher zur Bereitstellung der notwendigen Hintergrundinformationen, Lesegeräte zur Aufnahme von Schriften, Mikrophone zur akustischen Informationsaufnahme und Kameras zum optischen Erkennen interessierender Sachverhalte.

All dies hat der Anfänger normalerweise nicht, und deshalb soll in dieser grundlegenden Einführung auch vom Einsatz derartiger peripheren Geräte abgesehen werden.

Dies zwingt uns allerdings dazu, ein sehr schlichtes Beispiel zu wählen, das aber sehr wohl in der Lage ist, in anschaulicher Weise zu demonstrieren, welches die Grundprinzipien eines derartigen Programms sind.

Die Input-Informationen, die wir dem Rechner geben, und die dieser "erkennen" soll, sind schlichte Zahlen, genauer : mehrstellige, reelle Zahlen. Das BASIC-Programm, das wir dem Rechner eingeben, soll erkennen,

ob die Zahl positiv oder negativ ist,
ob sie reell oder ganzzahlig ist,
und aus welchen Symbolen sie sich zusammensetzt.

Geben wir also beispielsweise -12.65 ein, so soll das Programm folgende Ausgabe produzieren :

Die Zahl ist negativ;
die Zahl ist reell;
die Zahl lautet :
eins - zwei - Punkt - sechs - fünf

Daß es sich bei dieser sehr einfachen Aufgabenstellung, die der Leser aber leicht verkomplizieren kann (vergl. dazu Abschnitt 11.9), um ein echtes "Erkennens-Problem" handelt, wird aus der Problemanalyse des folgenden Abschnitts in Verbindung mit den Ausführungen der Abschnitte 11.1 und 11.2 deutlich.

11.4 Problemanalyse

Der Vorteil dieses Demonstrationsbeispiels liegt insbesondere darin, daß die Hintergrundinformationen, die der Rechner braucht, um die jeweils eingegebene Zahl erkennen zu können, sehr bescheiden sind : Es genügt nämlich dem Rechner die zehn Zahlworte des Dezimalsystems zu Vergleichszwecken zur Verfügung zu stellen.

Wenn dann die Input-Information vorliegt, muß das Programm zunächst feststellen, ob eine negative Zahl vorliegt oder nicht. Dafür bieten sich zwei Wege an :

Entweder man spaltet das erste Symbol ab und prüft, ob dieses "-" ist, oder man prüft einfach, ob die ganze Zahl kleiner ist als null.

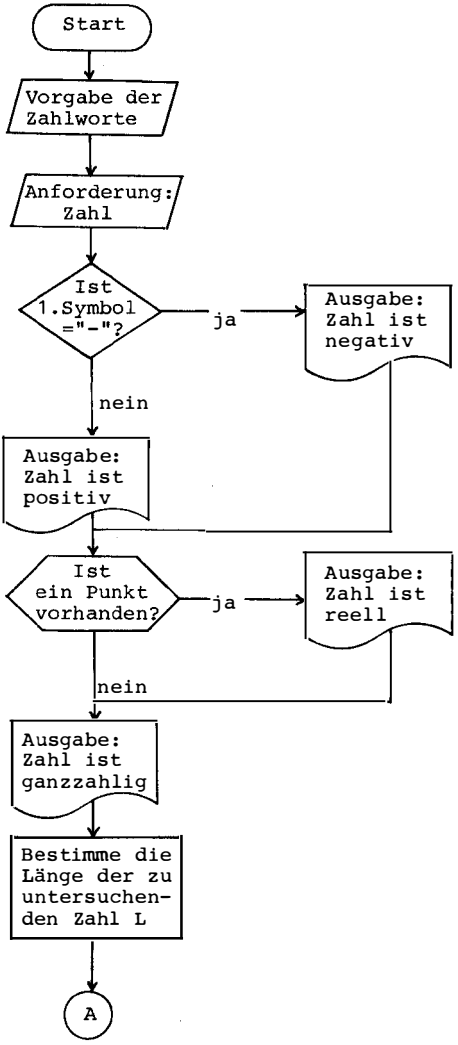
Die Feststellung, ob die Zahl reell ist oder ganzzahlig, wird getroffen, indem wir die Zahl mit der dafür geeigneten Stringbearbeitungsfunktion Symbol für Symbol darauf untersuchen, ob das Symbol "." vorliegt.

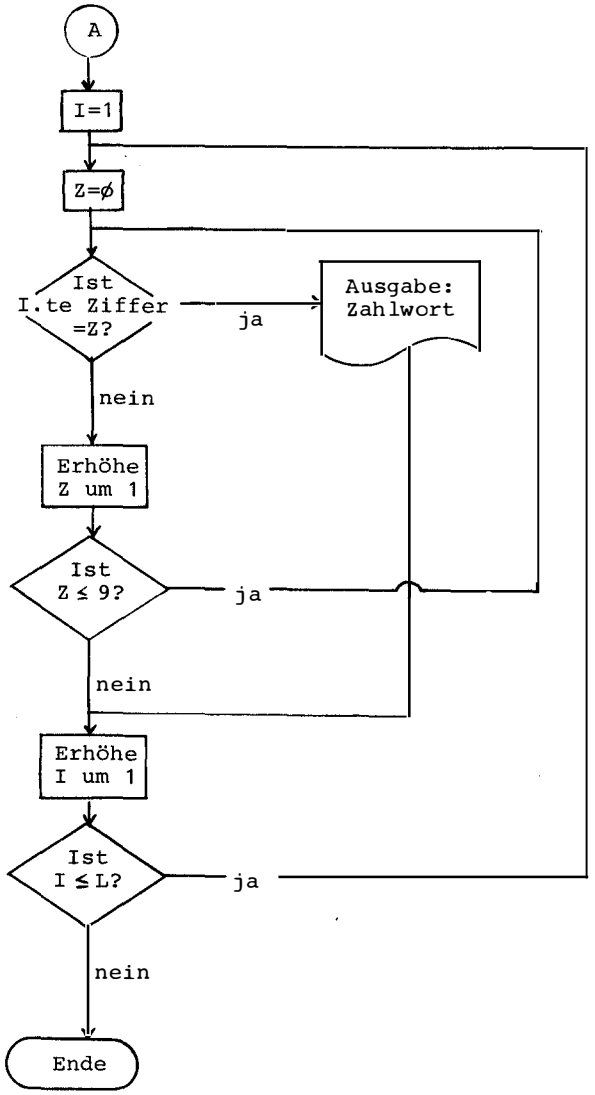
Die Ausgabe der einzelnen Ziffern in Worten schließlich erfordert das Abspalten jeder

einzelnen Ziffer und das Herausgreifen desjenigen Zahlworts aus den Hintergrundinformationen, das im Array der Zahlworte den gleichen Indexwert aufweist, wie er durch die jeweilige Ziffer vorgegeben wird.

Somit gelangen wir zu dem folgenden Programmablaufplan :

Flußdiagramm Erkennen




A
I=1
Z=ø
Ist
I.te Ziffer
=Z?
ja
Ausgabe:
Zahlwort
nein
Erhöhe
Z um 1
Ist
Z ≤ 9?
ja
nein
Erhöhe
I um 1
Ist
I ≤ L?
ja
nein
Ende

11.5 Programm

```
10 REM KI05 - ERKENNEN
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,     1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 5 :";
60 PRINT "   ERKENNEN"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM SIMULIERT   ERKENNEN":PRINT
90 PRINT"ZU DIESEM ZWECK SETZT ES EINGEGEBENE"
100 PRINT"ZAHLEN IN WORTE UM."
110 PRINT:PRINT:PRINT:PRINT
120 DIM Z$(10)
130 FOR I=0 TO  9:READ Z$(I):NEXT I
140 PRINT"BITTE EINE ZAHL EINGEBEN ":PRINT
150 PRINT"(MIT PUNKT, FALLS DEZIMALZAHL UND MIT"
160 PRINT"MINUSZEICHEN, FALLS NEGATIV)":PRINT
165 INPUT "IHRE ZAHL : ";A$
170 L=LEN(A$):DIM B$(L)
180 FOR I=1 TO L
190 B$(I)=MID$(A$,I,1)
200 IF B$(I)="-" OR B$(I)="." THEN 240
210 IF ASC(B$(I))<48 OR ASC(B$(I))>57 THEN GOTO  3000
240 NEXT I
```

```
250 PRINTCHR$(147)
260 IF B$(1)="-" THEN PRINT"ES IST EINE NEGATIVE ZAHL,":GOTO 280
270 PRINT"ES IST KEINE NEGATIVE ZAHL,"
280 FOR I=1 TO L
290 IF B$(I)="." THEN PRINT"UND ZWAR EINE DEZIMALZAHL.":GOTO 310
300 NEXT I
305 PRINT"UND ZWAR EINE GANZE ZAHL."
310 PRINT:PRINT
320 PRINTTAB(5)"DIE ZAHL LAUTET IN WORTEN :":PRINT:PRINT
325 A=1:IF B$(1)="-" THEN A=2
330 FOR I=A TO L
340 IF B$(I)="." THEN PRINT:PRINTTAB(3)"PUNKT":PRINT:GOTO 380
350 FOR Z= 0 TO 9
360 IF VAL(B$(I))=Z THEN PRINT TAB(5)Z$(Z):GOTO 380
370 NEXT Z
380 NEXT I
390 PRINT:PRINT:PRINT"ENDE DER AUSGABE":END


1000 REM DATEN
1010 DATA NULL,EINS,ZWEI,DREI,VIER
1020 DATA FUENF,SECHS,SIEBEN,ACHT,NEUN


2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN


3000 REM    FEHLER
3010 PRINT:PRINT:PRINT:PRINT
3020 PRINT"DIE EINGABE WAR FEHLERHAFT.":PRINT
3030 PRINT"BITTE NEUSTART MIT RUN" :PRINT:PRINT
3040 END
```

11.6 Variablenliste

A = Markiervariable
(Unterscheidung von negativen und positiven Zahlen)

A$ = Stringvariable
(Tastatureingabe und Input des Benutzers)

B$() = Symbole in der Eingabe-Information

I = Laufindex

L = Länge der eingegebenen Zahl

Z = Laufindex für Ziffern

Z$() = Zahlworte

11.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Abwarten

Satz 70-110 :

Erläuterungen zum Programm

Satz 120-130 :

Dimensionierung und Einlesen der Zahlworte
(Hintergrundinformation)

Satz 140-165 :

Anforderung der zu erkennenden Zahl mit erläuternden Hinweisen

Satz 170 :

Dimensionierung eines Arrays für die einzelnen Symbole nach Bestimmung der Anzahl der Symbole der eingegebenen Zahl

Satz 180 - 190 :

Beginn der Schleife zur Bestimmung der einzelnen Symbole und Abspaltung derselben (190)

Satz 200 :

Wenn das abgespaltene Symbol ein Minuszeichen oder ein Punkt ist, dann Sprung zum Satz 240, d.h. Bestimmung des nächsten Symbols; ist dies nicht der Fall, wird Satz 210 erreicht

Satz 210 :

Wenn das jeweils abgespaltene Symbol keine Ziffer ist (dann liegt seine ASCII-Codezahl unter 48 oder über 57), so erfolgt ein Sprung ins Unterprogramm 3000, um eine Fehlermeldung auszugeben

Satz 240 :

Beendigung der Prüfschleife über die einzelnen Symbole

Satz 250 :

Löschen des Bildschirms vor der Ausgabe der Ergebnisse des Erkennens

Satz 260 :

Ist das erste Symbol das Minuszeichen, so wird ausgegeben, daß es sich um eine negative Zahl

handelt und danach erfolgt ein Sprung zum Satz 280, wo auf Ganzzahligkeit geprüft wird

Satz 270 :

Ist dies nicht der Fall, wird ausgegeben, daß es sich nicht um eine negative Zahl handelt, und auch dann erfolgt die weitere Programmabarbeitung in Satz 280

Satz 280 - 300 :

In einer weiteren Schleife über alle Symbole wird nochmals untersucht, ob eines der Symbole ein Punkt ist; wenn ja, wird ausgegeben, daß es sich um eine reelle Zahl handelt, die Schleife wird verlassen und das Programm mit Satz 310 fortgesetzt

Satz 305 - 310 :

Wird hingegen die Schleife ganz über Satz 300 abgearbeitet, ohne daß der Hinaussprung zum Satz 310 erfolgte, dann deshalb, weil kein Punkt gefunden wurde, d.h. die Zahl ist ganzzahlig; dies wird ausgegeben

Satz 320 :

Ausgabe einer Überschrift für die weiteren Ergebnisse

Satz 325 :

Die Markiervariable A erhält den Wert 1 zugewiesen. Dies ist der Anfangswert der folgenden Ausgabeschleife (siehe Satz 330); dieser Anfangswert wird auf 2 erhöht, falls das erste Symbol das Minuszeichen war

Satz 330 - 380 :

Ausgabe der Zahlworte, es sei denn, das jeweilige Symbol ist ein Punkt; dann wird das Wort "PUNKT" ausgegeben und es erfolgt ein Sprung zum Satz 380,

d.h. es wird das nächste Symbol betrachtet (340);

das jeweilige Zahlwort, das zu dem gerade betrachteten Symbol, das jetzt eine Ziffer sein muß, paßt, wird in der Schleife (350-370) gefunden, indem die betrachtete Ziffer (VAL(B$(I))) mit dem Schleifenindex Z verglichen wird (360); ist dieser Vergleich erfolgreich, wird das entsprechende Zahlwort ausgegeben, andernfalls wird das nächste Zahlwort erprobt

Satz 390 :

Beendigung des Programms

Satz 1000 - 1020 :

Hintergrundinformationen in DATA-Statements (hier : Zahlworte)

Satz 2000 - 2040 :

Unterprogramm zum Abwarten einer Tastatureingabe (aus früheren Beispielen schon bekannt)

Satz 3000 - 3040 :

Unterprogramm zur Ausgabe einer Fehlermeldung, falls der Benutzer anstelle von Ziffern zum Beispiel Buchstaben eingegeben hat

11.8 Programmergebnisse

Wenn wir dieses Programm mit dem Kommando RUN starten, so fordert es vom Benutzer nach der Überschrift und einigen Erläuterungen die Zahl an, die erkannt werden soll.

Geben wir beispielsweise die Zahl -12.65 ein, so erhalten wir die folgenden Ergebnisse :

```
ES IST EINE NEGATIVE ZAHL,
UND ZWAR EINE DEZIMALZAHL.

     DIE ZAHL LAUTET IN WORTEN :

     EINS
     ZWEI

   PUNKT

     SECHS
     FUENF

ENDE DER AUSGABE
```

11.9 Ausblick

Das hier vorgestellte Programm liefert nur eine erste Vorstellung davon, was im Prinzip ein Programm leisten soll, das "erkennen" kann.

Eine Ausweitung auf optisches oder auf akustisches Erkennen ist, wie wir schon erläutert haben, an dieser Stelle nicht möglich.

Gleichwohl sind auch bei diesem einfachen Programm einige Ergänzungen vorstellbar, die es etwas anspruchsvoller machen. Der Leser könnte sich beispielsweise überlegen, wie dieses Programm verändert werden müßte, wenn die Ausgabe z.B. der Zahl 21 nicht in der Wortfolge "Zwei - Eins" bestehen soll, sondern in dem Wort "einundzwanzig".

Wenn er nach einigem Probieren diese Ergänzung programmieren kann, dann wird es ihm auch nicht zu schwer fallen, ein solches Programm noch komfortabler zu gestalten.

Kapitel 12 : Taktisches Spiel

12.1 Aufgabenstellung

Derjenige, der sich zum ersten Mal mit dem Thema künstliche Intelligenz befaßt, wird die zentralen Grundkonzepte, um die es in diesem Bereich des Computereinsatzes geht, vielleicht am ehesten am Beispiel taktischer Spiele erkennen können.

Bei solchen Spielen geht es darum, einen Computer in die Lage zu versetzen, angemessen auf Spielzüge des Benutzers reagieren zu können - mehr noch, solche Programme sollten dazu taugen, daß der Computer gegen den menschlichen Mitspieler das Spiel gewinnen kann.

Diese Zielsetzung erfordert taktische oder sog. strategische Überlegungen, was heißen soll, daß das Programm nicht einfach nach dem Zufallsprinzip seine Spielzüge auswählt, sondern zielgerichtet.

Eine der erstaunlichsten Entwicklungen in diesem Bereich stellen die teilweise außerordentlich leistungsfähigen Schachprogramme dar, die durchaus auch geübte Schachspieler in Verlegenheit bringen können.

Dieses Beispiel verdeutlicht, daß wir hier nicht über Zufallsspiele, wie zum Beispiel einfache Würfelspiele, und auch nicht über teilweise zufallsbeeinflußte Spiele, wie zum Beispiel Domino u.ä. sprechen wollen, sondern über solche Spiele, die (ohne Zufallseinfluß) determiniert sind.

Typische Vertreter dieser Art von Spielen sind das schon erwähnte Schach, aber auch zum Beispiel

Dame, Halma u.ä.

Es darf in diesem Zusammenhang nicht übersehen werden, daß viele intelligente Spielprogramme Einsatzbereiche abdecken können, die über das reine Unterhaltungsspiel am Familientisch weit hinausgehen : Man denke beispielsweise an die strategischen "Spiele" im Zusammenhang mit der Simulationen von Kriegsabläufen oder von politischen Verhandlungen, Bündnis- und Koalitionsbildungen usw.

Bei derartigen Ansätzen spielt aber wieder das hier ausgeschlossene Zufallselement in der Form eine Rolle, daß nicht alle "Spielparameter" bekannt sind, mithin Ungewißheiten bleiben, die nur mit Wahrscheinlichkeiten quantifiziert werden können, womit das Zufallselement wieder Eingang gefunden hätte.

Mit in diesen großen Bereich derjenigen Spiele, in denen sich determinierende Strategien und dazu Zufallselemente mischen, gehören die derzeit sehr aktuellen sog. Adventures , die den Bereich der Homecomputer sehr belebt haben.

12.2 Lösungsansätze

Die Erstellung intelligenter Spielprogramme ist im Prinzip sehr einfach, bei der praktischen Arbeit aber sehr kompliziert.

Denken wir noch einmal an das Schachspiel : Bekanntlich eröffnet Weiß die Partie und es steht außer Frage, daß der Spielbeginnende das Spiel gewinnen muß, wenn (allerdings nur dann) er keinen Fehler macht - selbst wenn Schwarz auch keinen Fehler begeht.

Allerdings gilt nun, daß diejenige Abfolge von Zügen, die fehlerfrei ist, mithin zum Spielgewinn führen muß, in diesem Sinne also die optimale Strategie darstellt, aus einer riesig großen Zahl von verschiedenen Spielzugfolgen (man spricht in diesem Zusammenhang von sog. Pfaden) ausgewählt werden muß.

Man halte sich nur einmal die folgende Überlegung vor Augen :

Wenn Weiß beginnt, so stehen ihm 20 verschiedene Eröffnungszüge zur Verfügung. Auf jeden dieser Eröffnungszüge kann Schwarz seinerseits mit jeweils 20 Gegenzügen antworten, so daß für das erste Zug-Paar schon 20*20 = 400 Möglichkeiten zur Verfügung stehen.

Danach ist wieder Weiß am Zug und hat nun wiederum rund zwei Dutzend Möglichkeiten usw., so daß wir nach zwei Zug-Paaren schon vor rund einer Viertel Million Möglichkeiten stehen.

Nun besteht aber ein Schachspiel nicht nur aus zwei, sondern aus vielleicht 15, 28 oder 60 Zug-Paaren, wodurch die Zahl der Pfade sich explosionsartig vervielfacht.

Kein Mensch, auch kein Schachweltmeister ist zu

Beginn einer Partie in der Lage, aus dieser fast unendlich großen Schar von Pfaden den jeweils spieloptimalen auszuwählen - und auch die Speicherkapazitäten der größten Rechner reichen nicht aus, um dieses Problem zu lösen, auch wenn es natürlich vom Grundsatz her lösbar ist.

Insoweit spielt doch auch bei diesem an sich determinierten Spiel der Zufall eine Rolle, und in der Tat passiert es ja ab und zu, daß Schwarz gewinnt.

Die Gesamtheit aller Pfade läßt sich anschaulich graphisch darstellen, indem man an jedem Spielzug alle die möglichen alternativen Antwortzüge des Spielgegners wie Zweige an einen Ast ansetzt. Auf diese Weise entsteht der sog. Entscheidungsbaum , der im Falle des Schachspiels in Milliarden und Abermilliarden von Zweigen endet.

Wenn allerdings bei einem bestimmten Spiel der Entscheidungsbaum so klein ist, daß für ihn die Speicherkapazität eines Rechners ausreicht, dann kann ein geeignetes Computerprogramm aus der Vielzahl von Verästelungen den optimalen Pfad heraussuchen und, sich an ihm entlangtastend, das Spiel gewinnen.

Wie gesagt, beim Schachspiel reichen die Kapazitäten heutiger Rechner dafür nicht. Selbst Großrechenanlagen sind "nur" in der Lage, bis 5 oder 6 Züge im Voraus zu planen und innerhalb dieser Spanne, innerhalb dieses Ausschnitts aus dem Gesamtbaum also, den teiloptimalen Pfad auszuwählen.

Insoweit ist ein solcher Computer schon den meisten Schachspielern überlegen, die allenfalls zwei bis drei Züge im Voraus planen können oder wollen.

Ist der Entscheidungsbaum zu groß, oder - was zu den gleichen Konsequenzen führt - sind bestimmte Äste nicht bekannt (verschwinden sie quasi im Nebel), dann muß wieder, wie schon bei Spielen,

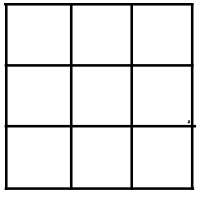
die zufallsbeeinflußt ablaufen sollen, mit Wahrscheinlichkeiten gearbeitet werden.

Mit Einzelheiten derartiger Fragestellungen, die für Roulette-, Lotto-, 17 und 4-, Poker- und Groschenautomatenspieler von Interesse sein dürften, wollen wir uns hier nicht beschäftigen und verweisen auf die einführende Literatur in die sog. Spieltheorie (siehe z.B.: MENGES,G. : 1969).

12.3 Das Beispiel

Zur Demonstration der Fragen, um die es hier geht, wählen wir ein überschaubares Beispiel eines determinierten Spiels, das sog. TIK-TAK-Spiel.

Das Spielfeld dieses Spiels besteht aus einem Quadrat aus drei Zeilen und drei Spalten gemäß der folgenden Skizze :



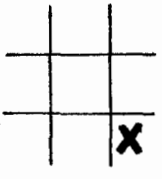
Spieler und Computer belegen nun in abwechselnder Folge je eines dieser neun Felder mit dem Symbol X (Spieler) bzw. O (Computer), wobei der Zufall darüber entscheidet, wer das Spiel beginnen darf.

Der Start wird beispielsweise durch einen Münzwurf festgelegt : Kommt Zahl, beginnt der Spieler, andernfalls beginnt der Computer.

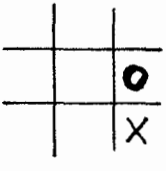
Das Spiel hat gewonnen, wer als erster eine komplette Zeile, oder eine komplette Spalte oder eine komplette Diagonale in dem 3*3-Feld mit seinem Symbol (also mit X oder mit O) besetzt hat (der Leser mag an das Mühle-Spiel erinnert sein, wo ähnliche Prinzipien zum Spielgewinn führen - wenn auch in mehreren Schritten).

Ein solches Spiel könnte folgendermaßen aussehen, wobei wir einmal unterstellen, daß der Spieler (also X) beginnen darf :

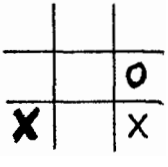
1. Zug :



2. Zug :



3. Zug :



4. Zug :

		O
X	**O**	X

5. Zug :

X		
		O
X	O	X

6. Zug :

X		
O		O
X	O	X

7. Zug :

X		
O	**X**	O
X	O	X

Wir erkennen : X (der Spieler) hat gegen O (gegen den Computer) gewonnen.

Es ist übrigens leicht einsichtig, daß derjenige einen beträchtlichen Chancenvorsprung hat, der dieses Spiel beginnen darf. Wenn er optimal spielt, also keinen Fehler macht, kann er nicht verlieren.

12.4 Problemanalyse

Um den Spielgewinn zu sichern, muß man - gemäß den obigen Überlegungen zum Begriff des Spielbaums - alle Zugfolgen analysieren, was voraussetzt, daß man sie erst einmal alle notiert.

Derjenige, der das Spiel beginnt, hat offenbar 9 Möglichkeiten, sein Symbol (X oder O) zu setzen. Der Gegenspieler hat dann als Reaktion noch 8 Möglichkeiten. Für das erste Zug-Paar erhalten wir somit 9*8 = 72 Möglichkeiten.

Im dritten Zug gibt es dann noch 7 Möglichkeiten, im vierten Zug noch 6 Möglichkeiten usw., so daß sich die Gesamtzahl der Zugfolgen, der Pfade also, ergibt zu :

Anzahl der Pfade = 9!

= 9*8*7*6*5*4*3*2*1 =

= 362 880

Wenn ein Kleinrechner all diese Pfade prüfen muß, benötigt er noch reichlich viel Rechenzeit (einige Stunden wird es wohl dauern), wogegen es für einen Großrechner kein Problem ist, die spieloptimale Zugfolge rasch zu finden.

Wir beschränken deshalb die strategischen Überlegungen unseres Homecomputers darauf, einen Sieg zu erzielen, wenn dies durch einen einzigen Zug, nämlich den nächstfolgenden möglich ist. Weiterhin soll das Programm in der Lage sein, einen Spielgewinn des Gegenspielers, also des Programmbenutzers zu verhindern, wenn dies mit dem nächstfolgenden Zug möglich ist.

Weitere strategische Regeln wollen wir in unser

Programm nicht einbauen, um die Rechenzeiten in überschaubaren Grenzen zu halten. Dies bedeutet, daß der Rechner zufällig setzt, wenn keine der beiden oben genannten Situationen vorliegt (Spielgewinn oder Verhinderung des Gewinns des Gegners).

Das hier zu entwickelnde Programm eignet sich besonders gut dazu, das Prinzip der strukturierten Programmierung (sog. Modul-Programmierung) zu demonstrieren, weil es bei dieser Problemstellung eine Reihe von Teilpoblemen gibt, die während des gesamten Programmablaufs mehrfach wiederkehren (beispielsweise die Ausgabe des Spielfeldes nach jedem Zug oder die Überprüfung, ob einer der Spielteilnehmer schon gewonnen hat usw.). Deshalb werden in größerem Umfang als bisher mehrfach Unterprogramme zur Erledigung solcher Aufgaben benutzt.

Im einzelnen sind unter problemanalytischen Gesichtspunkten die folgenden Teilprobleme zu lösen (der Leser vergleiche dazu auch das Flußdiagramm) :

1. Zunächst ist das Spielfeld (das 3*3-Quadrat; siehe obige Skizzen) auszugeben (im späteren Spielverlauf mit sich verändernder Besetzung).

Dies gelingt, indem wir durch POKE-Anweisungen unter Benutzung geeigneter Graphiksymbole der Commodore-Tastatur zwei senkrechte und zwei waagrechte Striche auf dem Bildschirm ausgeben und die enstehenden neun Quadratfelder dann mit POKE-Anweisungen sowohl mit den Ziffern 1 bis 9 (später dann mit X- und O-Symbolen) und danach die zu den entsprechenden Bildschirmspeicheradressen gehörenden Farbspeicheradressen mit Farbcodezahlen belegen.

Die erste Ausgabe dieses sog. Tableaus erfolgt in Standardfarben, d.h. hellblaue Ziffern und Striche auf blauem Grund.

Da dieses Tableau im weiteren Spielverlauf

mehrfach benötigt wird, werden wir die neun Bildschirmspeicheradressen über READ- und DATA-Statements einlesen. Daraus können wir durch Addition der Zahl 54272 die Farbspeicheradressen erzeugen.

Die neun Farbcodezahlen haben alle den Wert 14 (für hellblau) und die Felderbesetzungen sind die Ziffern 1 bis 9, die die Codezahlen 49 bis 57 aufweisen.

2. In einem zweiten Schritt muß geklärt werden, wer das Spiel beginnen darf, der Rechner oder der Programmbenutzer.

Diese Frage beantworten wir mit einer zufälligen Entscheidung. Unter Nutzung der RND-Funktion erzeugen wir eine ganzzahlige Zufallszahl 1 oder 2. Ist diese Zahl zufällig die Eins, so beginnt der Programmbenutzer, ist es die Zwei, so beginnt der Rechner das Spiel.

3. Nehmen wir einmal an, der Programmbenutzer sei am Zuge (ob Spielbeginn oder nicht, ist dabei jetzt nebensächlich). Damit er entscheiden kann, auf welches Feld er setzen soll, wird zunächst das jeweils gültige Tableau ausgegeben (siehe auch 1.), danach wird auf dem Bildschirm eine Liste derjenigen Felder ausgegeben, die überhaupt noch frei sind, also noch nicht mit einem X oder einem O besetzt sind.

Diese noch freien Felder durch das Programm finden und ausgeben zu lassen, ist relativ einfach, weil ja nur geprüft werden muß, welche der neun Felder noch nicht mit den Codezahlen 15 (für O) oder 24 (für X) besetzt sind.

Der Programmbenutzer wird dann aufgefordert, aus den noch freien Feldern, die auf dem Bildschirm ausgegeben wurden, eines auszuwählen und dieses wird dann mit dem Symbol X in weißer Farbe belegt. Dies geschieht einfach, indem die Adresse des Bildschirmspeichers des Feldes, das der Benutzer ausgewählt hat, mit der Codezahl 24 (für X) und

die Farbspeicheradresse mit der Codezahl 1 (für weiß) belegt wird.

Wenn der Spieler irrtümlicherweise ein Feld zur Besetzung auswählt, das schon besetzt ist, produziert das Programm eine Fehlermeldung und fordert erneut eine (nun hoffentlich korrekte) Besetzung an.

Wenn kein Feld mehr frei ist (und diese Situation erreicht wird, ohne daß der Rechner oder der Benutzer schon gewonnen hätte), dann ist das Spiel unentschieden zu Ende gegangen und eine ensprechende Schlußmeldung ist vorzusehen.

Wenn der Spieler seinen Zug getan hat, wird wiederum das Tableau ausgegeben (siehe 1.), um den gegenwärtigen Spielstand zu veranschaulichen.

4. Wenn der Benutzer (oder auch der Rechner) gesetzt hat, muß in einem nächsten Arbeitsschritt geklärt werden, ob das Spiel schon gewonnen ist.

Diese Überprüfung geschieht in der Weise, daß im Programm Zeile für Zeile und Spalte für Spalte und bei beiden Diagonalen überprüft wird, ob drei X-Symbole oder drei O-Symbole beisammen stehen. Ist dies für X-Symbole der Fall, so hat der Programmbenutzer gewonnen; ist dies hingegen für O-Symbole der Fall, hat der Computer gewonnen.

Damit dieses Unterprogramm sowohl verwendet werden kann, wenn es um die Frage geht, ob der Spieler gewonnen hat, als auch um die Frage, ob der Computer gewonnen hat, müssen die entsprechenden Zeilen-, Spalten- und Diagonalenüberprüfungen offen gehalten werden, in dem Sinn, daß vor Aufruf dieses Unterprogramms vorgegeben wird, ob auf die Codezahl 15 (für O) oder auf die Codezahl 24 (für X) überprüft wird.

5. Desweiteren muß der Programmteil gesondert betrachtet werden, der dafür zuständig ist, daß der Rechner seinen Spielzug macht.

In diesem Teil schlagen sich die (bescheidenen) strategischen Überlegungen des teilweise intelligenten Programms nieder :

Bevor der Rechner setzt, soll er zunächst selbst überprüfen, ob er gewinnen kann. Dies geschieht in der Weise, daß das Programm untersucht, ob in einer Zeile oder in einer Spalte oder in einer Diagonalen des 3*3-Tableaus schon zwei Symbole O stehen und gleichzeitig das jeweils dritte Feld noch frei ist.

Ist dies der Fall, so muß er sinnvollerweise das dritte noch offene Feld besetzen und hat dann gewonnen, was durch eine entsprechende Überprüfung bestätigt und ausgegeben wird (siehe 4.).

Ist dies nicht der Fall, so ist in einem zweiten Schritt zu überprüfen, ob das Gewinnen des Gegenspielers (des Programmbenutzers) verhindert werden muß.

Dies ist dann nötig, wenn in zwei Feldern einer Zeile oder einer Spalte oder einer Diagonalen schon X-Symbole stehen und das jeweils dritte Feld noch frei ist. Ist dies der Fall, muß das entsprechende dritte Feld mit dem Symbol O belegt werden.

Wenn der Rechner weder gewinnen kann noch das Gewinnen des Programmbenutzers verhindern muß, dann kann er frei aus den noch nicht besetzten Feldern auswählen. Zu diesem Zweck muß er untersuchen, welche Felder noch frei sind und aus den noch freien Feldern wählt er dann eines nach dem Zufallsprinzip, d.h. unter Verwendung der RND-Funktion aus.

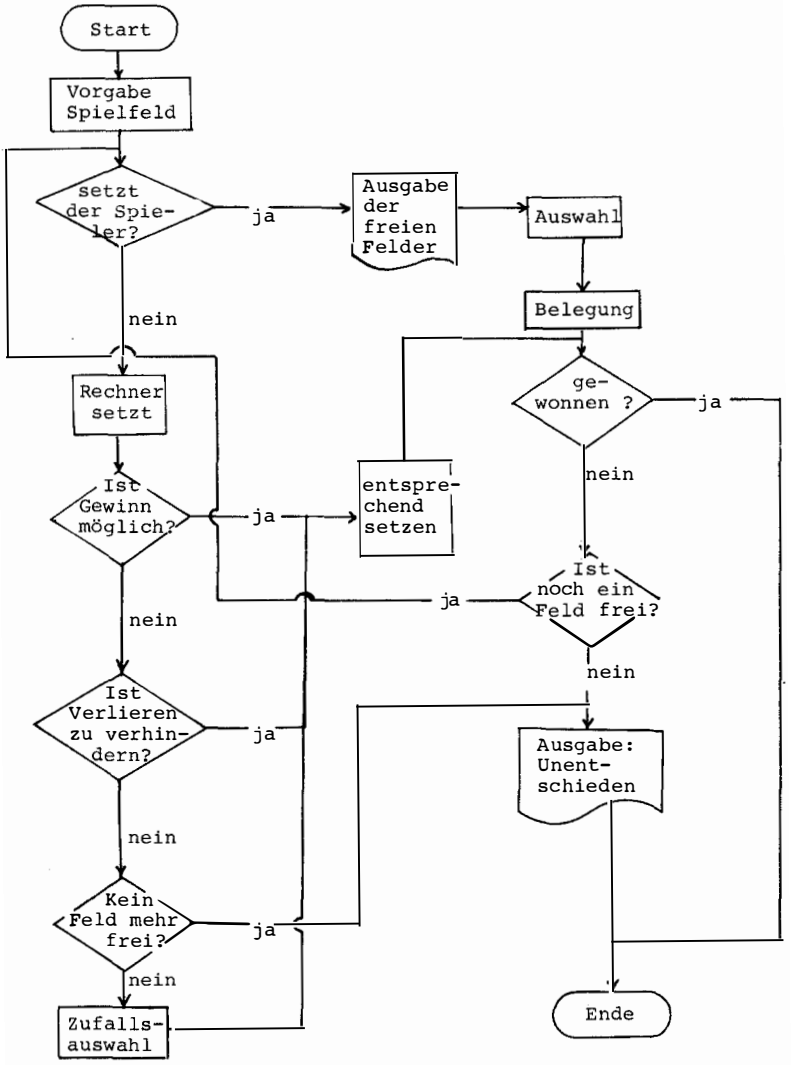
Ist kein Feld mehr frei, so ist das Spiel wieder unentschieden zu Ende gegangen.

Kann der Rechner hingegen setzen, so ist zu überprüfen, ob er nun seinerseits gewonnen hat. Dies geschieht nach dem gleichen Muster wie es oben schon beschrieben wurde.

Damit sind die wesentlichen Bausteine des Programms genannt. Man sieht, daß das Programm doch relativ aufwendig wird, obwohl die Problemstellung selbst doch offenbar nicht allzu anspruchsvoll ist.

Zu einem generellen Überblick verhilft der folgende Programmablaufplan (Flußdiagramm) :

Flußdiagramm TIK-TAK-Spiel



12.5 Programm

```
10 REM KI06 - TIK-TAK-SPIEL
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,    1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 6 :";
60 PRINT "TIKTAK-SPIEL"
70 E=10:GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM SPIELT DAS TIKTAK-SPIEL"
   :PRINT:PRINT
90 PRINT"SPIELTEILNEHMER SIND :":PRINT
100 PRINTTAB(5)"DER BENUTZER (ER SETZT ";
105 PRINT"⇶(";:PRINT"⧉)":PRINT
110 PRINT:PRINTTAB(5)"DER COMPUTER (ER SETZT ";
    :PRINT"■O";:PRINT"⧉)"
120 E=10:GOSUB 2000:REM WARTEN
122 DIM F(9),B(9),FA(9),C(9)
124 FOR I=1 TO 9:F(I)=I+48:NEXT I
126 FOR I=1 TO 9:READ B(I):FA(I)=B(I)+54272:NEXT I
128 FOR I=1 TO 9:C(I)=14:NEXT I
130 GOSUB 3000:REM TABLEAU
```

```
140 REM ZUFALLSSTART
150 R=INT(RND(1)*2+1)
160 IF R=1 THEN PRINTTAB(20)"SPIELER BEGINNT :":E=10:GOSUB
    2000:GOTO 230
170 PRINTTAB(20)"RECHNER BEGINNT :":PRINT
175 E=10:GOSUB 2000:REM WARTEN
177 PRINTTAB(20)"RECHNER SETZT":PRINT
180 GOSUB 4500:REM RECHNER SETZT
190 E=10:GOSUB 2000:REM WARTEN
230 GOSUB 4000:REM SPIELER SETZT
235 E=7:GOSUB 2000:REM WARTEN
240 GOTO 177
500 FOR I=1 TO 10:PRINT:NEXT I:PRINT"ENDE":END
1000 REM DATEN
1010 DATA 1065,1069,1073,1145,1149
1020 DATA 1153,1225,1229,1233
2000 REM UP WARTEN
2010 FOR I=1 TO E  :PRINT:NEXT I
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
3000 REM TABLEAU
3005 PRINTCHR$(147)
3010 FOR A=1104 TO 1114:POKE A,64:NEXT A
3020 FOR A=1184 TO 1194:POKE A,64:NEXT A
3030 FOR A=1067 TO 1264 STEP 40:POKE A,66:NEXT A
3040 FOR A=1071 TO 1231 STEP 40:POKE A,66:NEXT A
3050 FOR I=1 TO 9
3060 POKE B(I),F(I):POKE FA(I),C(I)
3070 NEXT I
3080 RETURN
```

```
4000 REM UP SPIELER SETZT
4002 GOSUB 3000:REM TABLEAU
4004 FOR I=1 TO 9:IF F(I)<>15 OR F(I)<>24 THEN 4007
4005 NEXT I
4006 GOTO 4125
4007 PRINTTAB(20)"SPIELER SETZT":PRINT
4010 PRINTTAB(20)"WELCHES FELD ?":PRINT:PRINTTAB(20)
     "AUSWAHL :":PRINT
4020 FOR I=1 TO 9
4030 IF F(I)=15 OR F(I)=24 THEN 4050
4040 PRINTTAB(20)I
4050 NEXT I
4060 PRINT:PRINTTAB(20)"BITTE ZAHL : ":PRINTTAB(20)"UND
     RETURN ";:INPUT Z
4070 IF F(Z)=15 OR F(Z)=24  THEN PRINT"FALSCHE EINGABE"
     :GOTO 4060
4080 REM BELEGEN
4090 FOR I=1 TO 9
4100 IF F(I)=15 OR F(I)=24 THEN 4120
4110 IF I=Z THEN F(I)=24:C(I)=1:GOTO 4130
4120 NEXT I
4125 FOR I=1 TO 7:PRINT:NEXT I:PRINTTAB(12)"UNENTSCHIEDEN."
4126 END
4130 GOSUB 3000:REM TABLEAU
4140 C=0:H=24:GOSUB 7000:REM GEWINNABFRAGE
4230 REM NOCH KEIN GEWINN
4240 FOR I=1 TO 9:IF F(I)<>15 AND F(I)<>24 THEN 4270
4250 NEXT I
4260 GOTO 4125
4270 RETURN
```

```
4500 REM UP RECHNER SETZT
4505 S=0
4510 REM GEWINNMOEGLICHKEIT
4530 H=15:G=24:J=15:M=0:GOSUB 6000
4550 IF S=1 THEN 5100
4700 REM VERLIEREN VERHINDERN
4710 H=24:G=15:J=15:M=0:GOSUB 6000
4730 IF S=1 THEN 5100
5000 REM RECHNER SETZT
5010 R=INT(RND(1)*9+1)
5020 IF F(R)=15 OR F(R)=24 THEN 5040
5030 F(R)=15:C(R)=0:GOTO 5090
5040 FOR I=1 TO 9:IF F(I)=15 OR F(I)=24 THEN 5060
5050 F(I)=15:C(I)=0:GOTO 5090
5060 NEXT I
5070 FOR I=1 TO 10:PRINT:NEXT I:PRINTTAB(12)"UNENTSCHIEDEN!"
5080 END
5090 GOSUB 3000:REM TABLEAU
5100 H=15:C=1:GOSUB 7000:REM GEWINNPRUEFUNG
5110 FOR I=1 TO 9:IF F(I)<>15 AND F(I)<>24 THEN 5140
5120 NEXT I
5130 GOTO 5070
5140 RETURN
```

```
6000 REM UP GEWINNMOEGLICHKEIT
6010 IF F(1)+F(2)=2*H AND F(3)<>G THEN F(3)=J:C(3)=M:GOTO 6250
6020 IF F(1)+F(3)=2*H AND F(2)<>G THEN F(2)=J:C(2)=M:GOTO 6250
6030 IF F(2)+F(3)=2*H AND F(1)<>G THEN F(1)=J:C(1)=M:GOTO 6250
6040 IF F(4)+F(5)=2*H AND F(6)<>G THEN F(6)=J:C(6)=M:GOTO 6250
6050 IF F(4)+F(6)=2*H AND F(5)<>G THEN F(5)=J:C(5)=M:GOTO 6250
6060 IF F(5)+F(6)=2*H AND F(4)<>G THEN F(4)=J:C(4)=M:GOTO 6250
6070 IF F(7)+F(8)=2*H AND F(9)<>G THEN F(9)=J:C(9)=M:GOTO 6250
6080 IF F(7)+F(9)=2*H AND F(8)<>G THEN F(8)=J:C(8)=M:GOTO 6250
6090 IF F(8)+F(9)=2*H AND F(7)<>G THEN F(7)=J:C(7)=M:GOTO 6250
6100 IF F(1)+F(4)=2*H AND F(7)<>G THEN F(7)=J:C(7)=M:GOTO 6250
6110 IF F(1)+F(7)=2*H AND F(4)<>G THEN F(4)=J:C(4)=M:GOTO 6250
6120 IF F(4)+F(7)=2*H ANO F(1)<>G THEN F(1)=J:C(1)=M:GOTO 6250
6130 IF F(2)+F(5)=2*H AND F(8)<>G THEN F(8)=J:C(8)=M:GOTO 6250
6140 IF F(2)+F(8)=2*H AND F(5)<>G THEN F(5)=J:C(5)=M:GOTO 6250
6150 IF F(5)+F(8)=2*H AND F(2)<>G THEN F(2)=J:C(2)=M:GOTO 6250
6160 IF F(3)+F(6)=2*H AND F(9)<>G THEN F(9)=J:C(9)=M:GOTO 6250
6170 IF F(3)+F(9)=2*H AND F(6)<>G THEN F(6)=J:C(6)=M:GOTO 6250
6180 IF F(6)+F(9)=2*H AND F(3)<>G THEN F(3)=J:C(3)=M:GOTO 6250
6190 IF F(1)+F(5)=2*H AND F(9)<>G THEN F(9)=J:C(9)=M:GOTO 6250
6200 IF F(1)+F(9)=2*H AND F(5)<>G THEN F(5)=J:C(5)=M:GOTO 6250
6210 IF F(5)+F(9)=2*H AND F(1)<>G THEN F(1)=J:C(1)=M:GOTO 6250
6220 IF F(3)+F(5)=2*H AND F(7)<>G THEN F(7)=J:C(7)=M:GOTO 6250
6230 IF F(3)+F(7)=2*H AND F(5)<>G THEN F(5)=J:C(5)=M:GOTO 6250
6240 IF F(5)+F(7)=2*H AND F(3)<>G THEN F(3)=J:C(3)=M:GOTO 6250
6245 REM SETZEN WAR NICHT MOEGLICH
6247 GOTO 6260
6250 S=1:GOSUB 3000
6260 RETURN
7000 REM GEWINNABFRAGE
7020 IF F(1)=H AND F(2)=H AND F(3)=H THEN 7120
7030 IF F(4)=H AND F(5)=H AND F(6)=H THEN 7120
7040 IF F(7)=H AND F(8)=H AND F(9)=H THEN 7120
7050 IF F(1)=H AND F(4)=H AND F(7)=H THEN 7120
7060 IF F(2)=H AND F(5)=H AND F(8)=H THEN 7120
7070 IF F(3)=H AND F(6)=H AND F(9)=H THEN 7120
7080 IF F(1)=H AND F(5)=H AND F(9)=H THEN 7120
7090 IF F(3)=H AND F(5)=H AND F(7)=H THEN 7120
7100 REM NOCH KEIN GEWINN
7110 RETURN
7120 FOR I=1 TO10 :PRINT:NEXT I
7125 IF C=1 THEN 7160
7130 PRINTTAB(10)"DU HAST GEWONNEN !"
7140 PRINT:PRINTTAB(7)"HERZLICHEN GLUECKWUNSCH !"
7150 END
7160 PRINTTAB(10)"ICH HABE GEWONNEN !"
7170 PRINT:PRINTTAB(13)"MEIN BEILEID"
7180 END
```

12.6 Variablenliste

A	=	Bildschirmspeicheradressen (Striche zeichnen im Tableau)
A$	=	Stringvariable zur Tastatureingabe
B()	=	Bildschirmspeicheradressen (Felder des Tableaus)
C	=	Markiervariable zur Kontrolle, ob der Rechner oder der Benutzer auf Gewinnmöglichkeit prüft
C()	=	Codezahlen für Felderfarben
E	=	Endwert der Leerzeilenschleife im Abwarteprogramm
F()	=	Codezahlen für die Felderbesetzungen
FA()	=	Farbspeicheradressen
G	=	Markiervariable zur Überprüfung von schon besetzten Feldern
H	=	wie G
I	=	Laufindex
J	=	Markiervariable zur Besetzung eines Feldes im Tableau
M	=	Markiervariable zur Farbänderung eines Feldes
R	=	Zufallszahl
S	=	Markiervariable zur Kontrolle, ob der Rechner gesetzt hat oder nicht
Z	=	Nummer des vom Spieler anzugebenden Feldes, das er zu besetzen wünscht

12.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tastatureingabe; die Aufforderung zur Betätigung einer Taste erfolgt nach Ausgabe von zehn Leerzeilen; um dies zu erreichen, wird E mit 10 belegt (vergl. auch Satz 2010)

Satz 80-110 :

Erläuterungen zum Programm

Satz 120 :

Wie Satz 70

Satz 122-128 :

Dimensionierungen (122), Erzeugen der Codezahlen für die Ziffern 1 bis 9 (124), danach Einlesen der Bildschirmspeicheradressen, Berechnung der Farbspeicheradressen (126) und Bestimmung der Codezahlen für die erste Farbgebung der Symbole in den Feldern (128)

Satz 130 :

Sprung ins Unterprogramm 3000 zur ersten Ausgabe des Tableaus

Satz 140-150 :

Erzeugung einer Zufallszahl 1 oder 2 zur Festlegung, wer das Spiel beginnen darf

Satz 160 :

Wenn der Zufallswert 1 erzeugt wird, darf der Spieler beginnen; dies erfordert nach dem Abwarten durch Sprung ins Unterprogramm 2000 einen Sprung zum Satz 230

Satz 170-177 :

Wenn der Spieler nicht beginnt, darf der Rechner beginnen; nach dem Sprung ins Unterprogramm 2000 zum Abwarten, erfolgt eine entsprechende Meldung

Satz 180 :

Sprung ins Unterprogramm 4500, in dem das Setzen durch den Rechner programmiert ist

Satz 190 :

Kehrt das Programm aus diesem Unterprogramm zurück, so wird wieder abgewartet (Unterprogramm 2000)

Satz 230 :

Nachdem der Rechner gesetzt hat, darf der Spieler setzen; deshalb Sprung ins Unterprogramm 4000; dieses Unterprogramm wird als erstes angesprungen, wenn der Spieler das Spiel eröffnen darf (siehe Satz 160)

Satz 235-240 :

Hat der Spieler gesetzt, wird wieder abgewartet (235) und durch Sprung zum Satz 177 (240) wird das nächste Zug-Paar in Gang gesetzt

Satz 500 :

Formale Beendigung des Hauptprogramms

Satz 1000-1020 :

Angabe der Bildschirmspeicheradressen für die neun Tableau-Felder

Satz 2000-2040 :

Unterprogramm zum Abwarten einer Tastatureingabe (dieses Unterprogramm wurde schon in früheren Beispielen beschrieben)

Neu ist der Satz 2010, der eine beliebige Anzahl von Leerzeilen, je nach Vorgabe von E, vor der Anforderung der Tastatureingabe ausgibt

Satz 3000-3080 :

Unterprogramm zur Ausgabe des Tableaus :

In Satz 3005 wird der Bildschirm gelöscht, in den Sätzen 3010-3040 werden zwei senkrechte und zwei waagrechte Striche gezeichnet; die Codezahl für den waagrechten Strich ist 64, die für den senkrechten Strich 66, die jeweiligen Positionen auf dem Bildschirm kann man auf kariertem Papier herausfinden;

in den Sätzen 3050 bis 3070 werden die Felder mit den zugehörigen Symbolen und die Adressen der Farbspeicher mit den zuständigen Farbcodezahlen besetzt

Satz 3080 bewirkt den Rücksprung ins rufende Programm

Satz 4000 :

Beginn des Unterprogramms, das die Feldbesetzung durch den Spieler (Programmbenutzer) ermöglicht

Satz 4002 :

Ausgabe des jeweiligen Tableaus

Satz 4004-4006 :

Überprüfung, ob wenigstens ein Feld im Tableau noch unbesetzt ist; ist dies der Fall, geht es bei Satz 4007 weiter; ist dies hingegen nicht der Fall, so erfolgt über Satz 4006 ein Sprung zum Satz 4125 (Ausgabe "Unentschieden")

Satz 4007-4010 :

Vorbereitung des Angebots an freien Feldern für den Spieler, damit dieser auswählen kann, welches er besetzen möchte

Satz 4020-4050 :

Ausgabe derjenigen Felder (4040), die noch nicht vom Symbol 0 oder vom Symbol X besetzt sind (Überprüfung in Satz 4030)

Satz 4060 :

Anforderung der Nummer desjenigen Feldes, das der Spieler zu besetzen wünscht

Satz 4070 :

Abfangen einer eventuellen fehlerhaften Eingabe und Rücksprung zum Satz 4060

Satz 4080-4120 :

Belegen des vom Spieler ausgewählten Feldes mit der Codezahl 24 (für X) und der Farbcodezahl 1 (für weiß) und überspringen der "Unentschieden" - Meldung durch Sprung zum Satz 4130

Satz 4125-4126 :

Der Satz 4125 mit der "Unentschieden"-Meldung wird nur erreicht, wenn kein freies Feld mehr vorhanden war (siehe Satz 4006); nach dieser Meldung kann das Programm dann beendet werden (4126)

Satz 4130 :

Wenn der Spieler gesetzt hat, wird das (nun veränderte) Tableau ausgegeben

Satz 4140 :

Durch Sprung ins Unterprogramm 7000 wird untersucht, ob der Spieler mit seinem letzten Zug vielleicht das Spiel gewonnen hat; zu diesem Zweck wird als Prüfgröße im Hilfsfeld H die Codezahl 24 (Symbol X) "mitgenommen"; die Markiervariable C wird auf 0 gesetzt, um erkennen zu können, daß nicht der Rechner am Zug war, wenn diese Überprüfung im Unterprogramm 7000 vorgenommen wird

Satz 4230 :

Kommt das Programm aus dem Unterprogramm 7000 zurück, dann deshalb, weil die Gewinnüberprüfung negativ ausgefallen ist (siehe auch Satz 7120)

Satz 4240-4270 :

In diesem Fall bleibt immer noch zu prüfen, ob der letzte Zug des Spielers vielleicht das Tableau gefüllt hat, ob er also gerade das letzte Feld besetzt hat, ohne damit einen Spielgewinn erzielt zu haben;

ist dies der Fall (Prüfung in Satz 4240), so erfolgt ein Sprung zum Satz 4125 (über Satz 4260), d.h. es kommt wieder zur "Unentschieden"-Meldung und das Programm wird beendet;

ist dies hingegen nicht der Fall, so kann das Spiel fortgesetzt werden, was durch den Rücksprung in das rufende Programm durch Satz 4270 geschieht;

damit ist dieses Unterprogramm beendet
Satz 4500-4595 :

Beginn des Unterprogrammms, welches das Setzen durch den Rechner bewerkstelligt;

um zu erkennen, daß der Rechner noch nicht gesetzt hat, haben wir die Markiervariable S mit dem Wert 0 belegt (4505)

Satz 4510-4530 :

Zunächst soll nun, gemäß unseren strategischen Überlegungen, der intelligente Rechner überprüfen, ob er mit dem ihm jetzt zustehenden Zug das Spiel für sich entscheiden kann;

dies gelingt, indem wir die Hilfsgrößen H, G, J und M wie in Satz 4530 belegen und dann ins Unterprogramm 6000 springen (siehe dort zur Erläuterung der Bedeutung dieser Hilfsgrößen)

Satz 4550 :

Wenn der Rechner aus diesem Unterprogramm 6000 zurückkommt und tatsächlich gesetzt hat, dann hat die Markiervariable S den Wert 1 (siehe Satz 6250);

an diesem Wert erkennen wir, daß keine weiteren strategischen Überlegungen mehr erforderlich sind (der Rechner hat dann ja in der Tat schon gesetzt) und deshalb erfolgt ein Sprung zum Satz 5100

Satz 4700 :

Hat der Rechner hingegen noch nicht gesetzt, dann gilt nach wie vor S=0 und deshalb wird dieser Satz 4700 erreicht

Satz 4710 :

Der Rechner muß jetzt überprüfen, ob er wenigstens mit seinem nächsten Zug das Verlieren des Spiels verhindern kann, indem er die Pläne des Spielers schlau durchkreuzt;

deshalb belegen wir jetzt die Hilfsfelder H, G, J und M anders und springen wieder ins gleiche Unterprogramm 6000

Satz 4730 :

Sollte nun der Rechner gesetzt haben, erfolgt die Rückkehr aus dem Unterprogramm 6000 wieder mit S=1 und deshalb wird wieder zum Satz 5100 gesprungen

Satz 5000 :

Dieser Satz wird nur dann erreicht, wenn der Rechner weder gesetzt hat, um zu siegen, noch, um den Sieg des Spielers zu verhindern; das heißt, daß er sich, unabhängig von strategischen Überlegungen, irgendein noch freies Feld zur Besetzung aussuchen kann

Satz 5010 :

Ermittlung einer Zufallszahl zwischen 1 und 9

Satz 5020-5030 :

Wenn das zufällig ausgewählte Feld schon mit einem O oder einem X besetzt sein sollte, erfolgt ein Sprung zum Satz 5040, andernfalls wird dieses Feld mit dem Rechnersymbol O in schwarzer Farbe (Codezahl 0) besetzt (5030); danach Sprung zum Satz 5090

Satz 5040-5060 :

Satz 5040 wird erreicht, wenn das zufällig ausgewählte Feld schon besetzt sein sollte; dann wird der Einfachheit halber das erste freie Feld durch den Rechner besetzt (5050); danach Sprung zum Satz 5090 (Überprüfung, ob noch ein Feld frei ist, in Satz 5040)

Satz 5070-5080 :

Wenn die Programmschleife in Satz 5040-5060 korrekt verlassen wird, ohne daß der Rechner setzen kann, dann deshalb, weil kein freies Feld mehr vorhanden war;

dies bedeutet, daß das Spiel beendet ist, ohne daß einer der Spielteilnehmer gewonnen hätte, weshalb die Ausgabe "unentschieden" erfolgt (5070) und das Programm beendet wird (5080)

Satz 5090 :

Dieser Satz wird nur erreicht, wenn der Rechner setzen konnte; um das Ergebnis dieses Setzens zu veranschaulichen, wird jetzt das veränderte Tableau ausgegeben

Satz 5100 :

Jetzt muß überprüft werden, ob der Rechner nach seinem letzten Zug gewonnen hat; dies geschieht durch Sprung ins Unterprogramm 7000, wobei jetzt die Hilfsgrößen H=15 und C=1 "mitgenommen" werden müssen (siehe dort zur Erklärung)

Satz 5110-5140 :

Kehrt der Rechner aus dem Unterprogramm 7000 an diese Stelle zurück, dann deshalb, weil er noch nicht gewonnen hat; es muß deshalb nun überprüft werden, ob überhaupt noch ein Feld nach seinem letzten Zug frei ist;

ist dies nicht der Fall, so erfolgt über Satz 5130 der Sprung zum Satz 5070 ("Unentschieden"-Meldung

und Beendigung des Programms), andernfalls erfolgt über Satz 5140 der Rücksprung ins rufende Programm;

damit ist auch dieses umfangreiche Unterprogramm abgeschlossen
Satz 6000 :

Mit Satz 6000 beginnt das Unterprogramm, welches prüft, ob der Rechner das Spiel gewinnen kann;

dieses Unterprogramm kann gleichermaßen verwendet werden, um zu prüfen, ob der Spielgewinn durch den Spielgegner verhindert werden kann;

die erste Frage wird so geprüft, daß die Hilfsgröße H den Wert 15, die Hilfsgröße G den Wert 24, die Hilfsgröße J den Wert 15 und die Hilfsgröße M den Wert 0 mitbringt (vergl. Satz 4530); zur Erläuterung siehe Satz 6010 :

Satz 6010 :

Der Rechner prüft, ob die Felder 1 und 2 mit den Codezahlen 15 besetzt sind (die Summe der Codezahlen muß dann 2*H = 2*15 = 30 sein), ob also beide Felder schon das Symbol 0 tragen und ob gleichzeitig das dritte Feld noch nicht mit dem Symbol X besetzt ist;

wenn dies der Fall ist, dann besetzt er das dritte Feld mit J=15 (Symbol 0) und die Farbspeicheradresse C(3) mit 0 (schwarz);

die erste Zeile des Tableaus ist dann mit drei Symbolen 0 gefüllt (der Rechner hat gewonnen!) und es erfolgt ein Sprung zum Satz 6250, d.h. die Markiervariable S wird mit dem Wert 1 belegt (der Rechner hat gesetzt), das Tableau wird ausgegeben (Sprung ins Unterprogramm 3000) und es erfolgt der Rücksprung ins rufende Programm (6260);

genauso kann in diesem Satz überprüft werden, ob der Gewinn des Gegners verhindert werden soll;

dies ist nämlich dann erforderlich, wenn die Summe der ersten beiden Feldercodes = 2*H = 2*24 = 48 ist (dann stehen dort die Symbole X, dessen Codezahl 24 ist). Deshalb bringt jetzt die Hilfsgröße H den Wert 24 mit (siehe Satz 4710);

ist dies der Fall und gleichzeitig das dritte Feld nicht mit dem Symbol O belegt (G bringt die Codezahl 15 mit; siehe 4710), dann wird es mit dem Symbol O belegt (J bringt die Codezahl 15 mit) in der Farbe schwarz (M bringt die Codezahl 0 mit);

auch dann erfolgt der Sprung zum Satz 6250

Satz 6020-6240 :

In diesen Sätzen geschieht das gleiche, wie ausführlich eben für Satz 6010 beschrieben wurde, nun aber für die übrigen Zeilen, für die drei Spalten und für die beiden Diagonalen des 3*3-Tableaus;

jede Zeile, jede Spalte und jede Diagonale macht dabei drei derartige Abfragen erforderlich :

In der ersten Zeile beispielsweise müssen die Felder 1 und 2 (s.o.), 1 und 3 bzw. 2 und 3 überprüft werden; entsprechend für die anderen Zeilen, für die Spalten und für die Diagonalen

Satz 6245-6247 :

Dieser Satz wird nur erreicht, wenn bei keiner der insgesamt 24 Überprüfungen zum Satz 6250 gesprungen werden konnte;

ist dies der Fall, so wird über Satz 6247 zu Satz 6260 gesprungen, d.h. es erfolgt der Rücksprung ins rufende Programm, ohne daß gesetzt wurde

Satz 6250-6260 :

Dieser Satz wird nur erreicht, wenn gesetzt werden

konnte; deshalb wird die Markiervariable S mit dem Wert 1 belegt, das veränderte Tableau wird ausgegeben und erst dann erfolgt über Satz 6260 der Rücksprung ins rufende Programm,

Damit ist auch dieses Unterprogramm, das die bescheidene Strategie des Rechners bei diesem Spiel steuert, beendet
Satz 7000 :

Mit diesem Satz beginnt das Unterprogramm, welches prüft, ob das Spiel schon gewonnen ist;

wenn dies der Fall sein sollte, muß eine Zeile oder eine Spalte oder eine Diagonale der 3*3-Matrix mit gleichen Symbolen gefüllt sein;

bringt die Hilfsgröße H den Wert 15 mit, wird also geprüft, ob diese Bedingung für den Rechner erfüllt ist (Symbol O), bringt H hingegen den Wert 24 mit, so wird geprüft, ob diese Bedingung für den Programmbenutzer erfüllt ist (Symbol X)

Satz 7020-7090 :

Ist diese Bedingung ein einziges Mal erfüllt - dies wird in diesen Sätzen überprüft - so springt das Programm zum Satz 7120

Satz 7100-7110 :

Diese Sätze werden nur erreicht, wenn die Gewinn-Bedingung in keinem Fall erfüllt ist; deshalb erfolgt über Satz 7110 der Rücksprung ins rufende Programm

Satz 7120 :

Ausgabe von 10 Leerzeilen

Satz 7125 :

Wenn C den Wert 1 hat, dann deshalb weil der Rechner den letzten Zug tat; es erfolgt dann ein Sprung zum Satz 7160

Satz 7130-7150 :

Satz 7130 wird erreicht, wenn die Gewinnbedingung erfüllt war und der Spieler den letzten Zug tat, d.h. der Spieler hat gewonnen;

dies wird ausgegeben und das Programm mit dem Satz 7150 beendet

Satz 7160-7180 :

Satz 7160 wird erreicht, wenn der Rechner den letzten Zug tat (siehe 7125), d.h. er hat gewonnen;

dies wird ausgegeben und das Programm mit Satz 7180 beendet;

damit ist auch dieses Unterprogramm und somit auch das gesamte Programm beendet.

12.8 Programmergebnisse

Die Darstellung der Ergebnisse dieses Programms sind entbehrlich. Es ergibt sich nach Starten des Programms ein Ablauf, wie er in einem der ersten Abschnite dieses Kapitels anhand einer Skizze anschaulich beschrieben wurde.

12.9 Ausblick

Wir sehen an dem obigen Beispiel, daß selbst recht einfache strategische oder halb-strategische Spiele zu recht umfangreichen Programmen führen.

Man kann sich deshalb sicherlich leicht vorstellen, daß kompliziertere Spiele einen beträchtlichen Programmieraufwand und, was noch wichtiger ist, große Rechenzeiten beanspruchen.

Es ist deshalb in der Regel sicher angebracht, umfangreichere Spiele in einer maschinennäheren Sprache zu programmieren (zum Beispiel in ASSEMBLER), was dann die Rechenzeiten deutlich verringert.

In den einleitenden Bemerkungen wurde schon darauf hingewiesen, daß viele Spiele reine Zufallsspiele sind. Bei derartigen Spielen stehen aber andere Programmelemente als sie hier vorgestellt wurden im Vordergrund.

Das folgende Beispiel zeigt ein einfaches Würfelspiel, dessen Beschreibung sich der Leser übungshalber selbst erarbeiten sollte :

```
10 REM WUERFELSPIEL
20 PRINTCHR$(147)
30 W=INT(RND(1)*6+1)
40 PRINT"ICH WUERFELE EINE ";W
50 PRINT:PRINT:PRINT
60 PRINT"WAS WUERFELST DU ?"
70 PRINT:PRINT:PRINT
75 PRINT"BITTE EINE TASTE DRUECKEN !"
76 GET B$: IF B$="" THEN 76
77 PRINT:PRINT:PRINT
80 V=INT(RND(1)*6+1)
90 PRINT"DEIN WURF IST EINE ";V
100 PRINT:PRINT:PRINT
110 IF W>VTHEN PRINT"ICH HABE GEWONNEN":GOTO 140
120 IF W<V THEN PRINT"DU HAST GEWONNEN":GOTO 140
130 PRINT"UNENTSCHIEDEN !"
140 PRINT:PRINT:PRINT
150 INPUT "NOCHMAL (J/N) ";A$
160 IF A$="J" THEN 20
170 PRINT:PRINT:PRINT"ENDE":END
```

Kapitel 13 : Selbstlernende Programme

13.1 Aufgabenstellung

Das Beispiel des letzten Kapitels hat dem Leser anschaulich gezeigt, daß man die Gewinnchancen des Rechners dann wesentlich verbessern könnte, wenn es gelingt, ihn in die Lage zu versetzen, sich erfolgreiche Spielzüge zu merken, um diese dann bei folgenden Spielen wieder einzusetzen.

Genauso wäre vorstellbar, daß er Fehler, die dazu geführt haben, daß er sein Spiel verliert, in Zukunft nicht mehr begeht, aus Schaden also klüger wird.

Es ist einzusehen, daß sich die Leistungsfähigkeit eines Rechners in derartigen Spielprogrammen auf diese Weise wesentlich verbessern ließe.

Das gleiche gilt in entsprechender Weise auch für Suchsysteme und Auskunftsprogramme oder für Expertensysteme : Wenn das jeweilige Programm eine fehlerhafte Auskunft gibt, dann könnte dafür gesorgt werden, daß es sich die anschließende Korrektur merkt, so daß es den gleichen Fehler niemals mehr begeht.

Denkbar wäre auch, daß ein solches Programm so angelegt wird, daß es - indem es ständig weitere Fragen an den Benutzer stellt - seine eigene Informationsbasis fortwährend verbessert und auf diese Weise immer klüger wird.

Der Rechner lernt hinzu - wenn wir geeignete Programme einsetzen.

Ein letztes typisches Beispiel kann diese Aufgabe

sehr deutlich illustrieren :

Man stelle sich einen programmgesteuerten Roboter vor, der den Ausgang eines Labyrinths suchen soll. Wenn er schließlich nach langem Suchen diesen Ausgang gefunden hat, kann er sich, wenn er über das entsprechende Progamm verfügt, sowohl die erfolgreichen Wegstrecken, wie auch jene, die in die Irre geführt haben, gemerkt haben. Ist dies der Fall, wird er in einem zweiten Versuch sofort und ohne Umwege den richtigen Weg finden.

Wenn er auf diese Weise fortwährend seine Informationsbasis verbessert, dann kann er auch schon in der ersten Suche seinen Weg optimieren, indem er Irrwege niemals ein zweites Mal begeht.

Derartige selbstlernende Programme sind bei Experten- und Auskunftssystemen inzwischen von großer praktischer Bedeutung, weil sie natürlich den Nutzen derartiger Systeme ständig verbessern.

Deshalb soll auch zu diesem Fragenbereich hier ein Programmbeispiel vorgestellt werden, wobei wir uns nun aber, nach den umfangreichen Beispielen der vorangegangenen Kapitel, sowohl was das Beispiel selbst, aber auch was die Problemanalyse und Programmbeschreibung betrifft, etwas kürzer fassen können; dies insbesondere auch deshalb, weil unter programmlogischen Gesichtspunkten keine wesentlich neuen Elemente hinzutreten.

13.2 Lösungsansätze

Der grundsätzliche Lösungsansatz bei der Entwicklung selbstlernender Programme besteht darin, daß

1. beim Programmablauf zusätzliche Informationen durch das Programm selbst angefordert werden oder daß sie unabhängig von derartigen Anforderungen vom Benutzer eingegeben werden können,

2. diese zusätzlichen Informationen in Ergänzung oder auch vielleicht teilweisen Ersetzung der bisherigen Informationsbasis gespeichert werden,

3. die zusätzlichen Informationen im passenden Sachzusammenhang auch wieder vom Programm selbst abgerufen und dem Benutzer bereitgestellt werden können.

Eine derartige Programmstruktur setzt voraus, daß das Programm möglichst von sich aus die Lücken im Informationsbestand erkennt, die vom Benutzer dann geschlossen werden sollen und eventuell auch geschlossen werden können.

13.3 Das Beispiel

Ein ganz typisches Beispiel für ein derartiges selbstlernendes Programm, das in ähnlicher Form auch unter dieser Überschrift ab und zu in der Literatur und in BASIC-Programmsammlungen auftaucht, ist das Programm "Tiere raten".

Bei diesem Beispiel geht es darum, daß sich der Programmbenutzer ein Tier ausdenkt, daß der Rechner dank seines Programms zu raten versucht. Der Leser erkennt sicherlich die Parallele zu dem früher besprochenen Expertensystem :

Bei einem solchen System, das beispielsweise aus dem medizinischen Bereich stammt, "denkt" sich der Patient eine Krankheit (d.h. er hat sie) und das Expertensystem versucht, diese Krankheit zu "erraten", d.h. die richtige Diagnose zu stellen.

Dies geschieht, indem durch ständig weitere Informationsanforderungen das Wissen des Systems verbessert wird, um zu einer möglichst präzisen Beschreibung des Krankheitsbildes zu gelangen und dann eine möglichst treffsichere Diagnose stellen zu können.

In unserem Beispiel besteht die Ausgangssituation darin, daß der Rechner nur zwei Tiere beim Start des Programms kennt. Diese beiden Tiere sind "Fisch" und "Vogel".

Darüberhinaus kennt der Rechner eine Frage, mit deren Hilfe man in der Lage ist, diese beiden Tiere voneinander zu unterscheiden. Diese Frage lautet : "Kann das Tier schwimmen ?"

Wenn der Benutzer sich nun "Fisch" als Tier gemerkt haben sollte, dann wird er auf die obige Frage, die ihm im Programmablauf gestellt wird, mit "Ja" antworten müssen, und an diesem "Ja" erkennt der Rechner, daß sich der Benutzer "Fisch"

gemerkt hat. Antwortet der Benutzer hingegen mit "Nein", dann geht der Rechner davon aus, daß "Vogel" das zu erratende Tier war.

Nun kann es aber natürlich sein, daß der Benutzer sich weder "Fisch" noch "Vogel", sondern vielleicht "Maus" gemerkt hat. Er antwortet deshalb auf die Frage, ob das Tier schwimmt, mit "Nein" und auch auf die Feststellung, daß es sich doch dann sicher um einen Vogel handeln müsse, ebenfalls mit "Nein".

Damit ist das Programmsystem in die Situation geraten, daß es seinen Informationshintergrund verbessern muß und auch kann.

Um dies zu erreichen, fragt es den Benutzer, an welches Tier er denn gedacht hätte. Der Benutzer antwortet wahrheitsgemäß "Maus" und dieses Tier merkt sich nun das Programm durch Speicherung. Es kennt jetzt also drei Tiere, nämlich "Fisch", "Vogel" und "Maus".

Diese Information allein nützt aber zukünftig noch nichts, wenn zusätzlich nicht noch eine zweite Frage in den Informationsbestand aufgenommen wird. Diese Frage muß in der Lage sein, so wie die erste zwischen "Fisch" und "Vogel" erkennensmäßig trennen konnte, zwischen "Vogel" und "Maus" zu trennen.

Diese Frage wird deshalb ebenfalls vom Benutzer angefordert und könnte zum Beispiel folgendermaßen lauten :

"Kann das Tier fliegen ?"

Wenn nun das Ratespiel erneut gestartet wird, verfügt das System über einen verbesserten Informationshintergrund : Es kennt schon drei Tiere und kann durch zwei geeignete Fragen eine Trennung zwischen ihnen herbeiführen. Es hat also eine wesentlich verbesserte Trefferchance beim Raten.

Sollte der Rechner aber auch im zweiten Spiel nicht in der Lage sein, das ausgedachte Tier (vielleicht jetzt "Katze") zu erraten, dann wird er gemäß den obigen Überlegungen im nächsten Spiel schon vier Tiere kennen ("Fisch", "Vogel", "Maus" und "Katze") und drei Fragen zur Trennung zwischen denselben, nämlich :

"Kann das Tier schwimmen ?"
"Kann das Tier fliegen ?"
"Kann das Tier schnurren ?"

Dies bedeutet, daß von Spielrunde zu Spielrunde der Rechner klüger wird, und somit die Chance immer größer wird, daß er das ausgedachte Tier erraten kann.

Die Grundidee eines solchen Programms besteht also darin, daß entweder das Programm eine zutreffende Antwort gibt, oder aber daß es zumindest von Versuch zu Versuch klüger wird.

13.4 Problemanalyse

Die Ausführungen des vorangegangenen Abschnitts haben schon die wesentlichen Stichworte zur Problemanalyse geliefert, so daß wir uns hier auf einige wenige Ergänzungen und auf die Präsentation eines generellen Flußdiagramms beschränken können.

Als Ausgangsinformationen geben wir zwei Tiere und eine sie unterscheidende Frage vor, um dann den Programmbenutzer durch das Programm aufzufordern, an ein bestimmtes Tier zu denken.

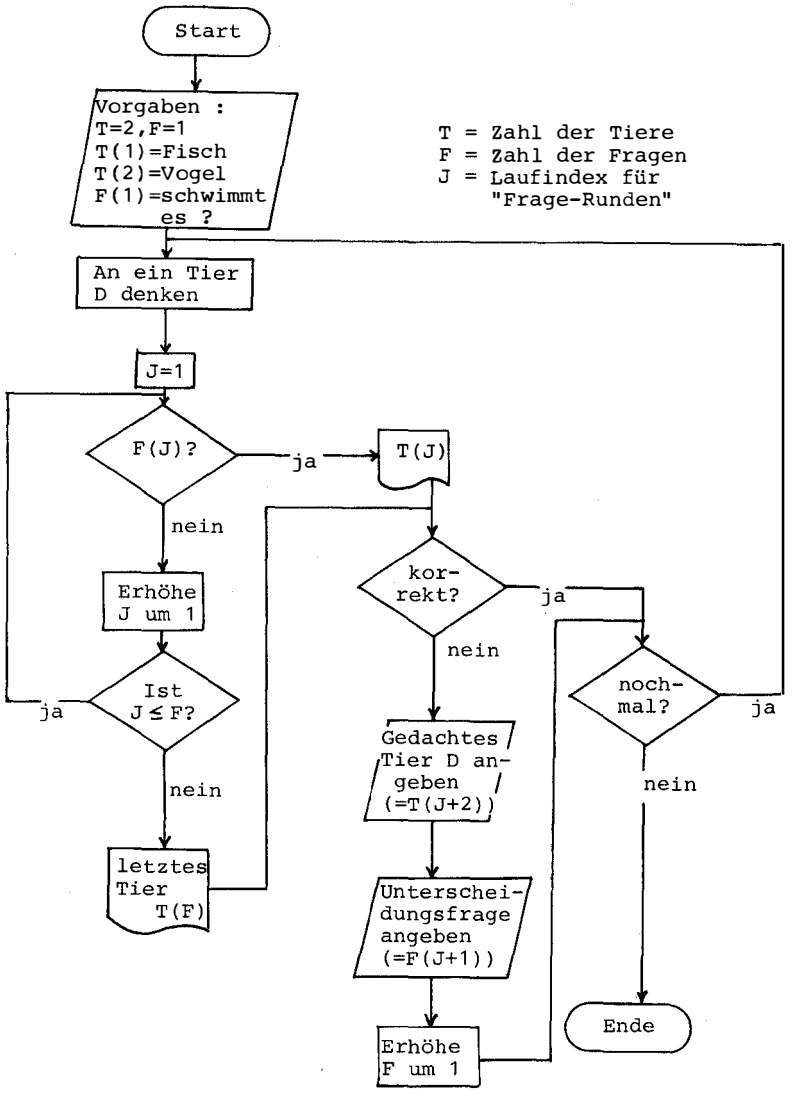
Der Rechner stellt daraufhin die eine einzige Frage, die ihm bislang zur Verfügung steht, um entscheiden zu können, ob er das eine oder das andere Tier aus seinem Informationsvorrat als "Rateergebnis" präsentiert.

Wenn beide Alternativen nicht zutreffen, so fordert er zusätzliche Informationen an, nämlich den Namen des Tiers, das sich der Benutzer gedacht hat, und das der Rechner nicht erraten konnte, und darüberhinaus eine Frage, die es erlaubt, dieses Tier von dem letzten, das er als (falsches) Rateergebnis angeboten hatte, zu unterscheiden.

Danach kann ein neues Spiel in Angriff genommen werden.

Als Flußdiagramm stellt sich dieser Programmablauf folgendermaßen dar :

Flußdiagramm "Tiere raten"



13.5 Programm

```
10 PRINTCHR$(147)
20 PRINTTAB(7)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
30 PRINTTAB(7)"PROF.DR.W.VOSS,     1985"
35 PRINT:PRINT:PRINT:PRINT
40 PRINTTAB(7)"BEISPIEL  8  : "
45 PRINTTAB(7)"SELBSTLERNENDES  SYSTEM":PRINT:PRINT
47 PRINT:PRINT:PRINTTAB(12)"BITTE WARTEN"
48 FOR J=1 TO 7000:NEXT J
49 PRINTCHR$(147)
50 PRINTTAB(5)"DENKE AN EIN BESTIMMTES TIER"
55 PRINTTAB(5)"UND DER COMPUTER WIRD VERSU-"
60 PRINTTAB(5)"CHEN, DIESES TIER ZU RATEN !"
70 DIM A$(200)
80 FOR I=0 TO 3:READ A$(I):NEXT I
90 N=VAL(A$(0))
120 PRINT:PRINT:PRINT
122 PRINT"WILLST DU VOR DEM NAECHSTEN SPIEL WIS-"
123 PRINT"SEN, WELCHE TIERE ICH SCHON KENNE ?"
124 PRINT:INPUT "JA (J) ODER NEIN (N) ";A$
125 IF A$="J" THEN 600
126 PRINT:PRINT:PRINT
130 INPUT "DENKST DU AN EIN TIER (J/N) ";A$
150 IF A$<>"J" THEN 120
160 K=1
165 PRINT:PRINT:PRINT
170 GOSUB 390:REM FRAGEN
180 IF LEN(A$(K))=0 THEN 999
190 IF LEFT$(A$(K),2)="/Q" THEN 170
195 PRINT:PRINT:PRINT
200 PRINT"HEISST DAS TIER ";RIGHT$(A$(K),LEN(A$(K))-2);
210 INPUT" (J/N) ";A$
220 IF A$="J" THEN G=1:GOTO 700
238 PRINT:PRINT:PRINT"GIB BITTE AN, AN WELCHES TIER DU GE-"
240 PRINT"DACHT HAST ";:INPUT V$
245 PRINT:PRINT:PRINT CHR$(147)
250 PRINT"BITTE GIB EINE FRAGE EIN, BEI DEREN"
252 PRINT"BEANTWORTUNG MIT J (FUER JA) ODER N"
254 PRINT"(FUER NEIN) SICH "
255 PRINT:PRINTTAB(5)V$;"  UND  ";RIGHT$(A$(K),LEN(A$(K))-2):PRINT
256 PRINT"VONEINANDER UNTERSCHEIDEN.":PRINT:PRINT
260 PRINT:PRINT"BEISPIEL : KANN DAS TIER BEISSEN":PRINT:PRINT
270 INPUT "FRAGE  : ";X$
274 PRINT:PRINT:PRINT
```

```
275 PRINT"ANTWORTE MIT J (JA) ODER N (NEIN)":PRINT:PRINT
277 PRINT:PRINT:PRINT
280 PRINT"BEI DER TIERART ";V$;" MUSS ICH"
285 INPUT "ANTWORTEN (J/N) ";A$
300 IF A$<>"J" AND A$<>"N" THEN 280
310 IF A$="J" THEN B$="N"
320 IF A$="N" THEN B$="J"
330 Z1=VAL(A$(0))
340 A$(0)=STR$(Z1+2)
350 A$(Z1)=A$(K)
360 A$(Z1+1)="/A"+V$
370 A$(K)="/Q"+X$+"/"+A$+STR$(Z1+1)+"/"+B$+STR$(ZI)+"/"
380 GOTO 120
390 REM UP FRAGEN
395 PRINT:PRINT:PRINT
400 Q$=A$(K)
410 FOR Z=3 TO LEN(Q$)
415 IF MID$(Q$,Z,1)<>"/" THEN PRINTMID$(Q$,Z,1);:NEXT Z
420 PRINT" (J/N) ";:INPUT C$
440 IF C$<>"J" AND C$<>"N" THEN 410
450 T$="/"+C$
455 FOR X=3 TO LEN(Q$)-1
460 IF MID$(Q$,X,2)=T$ THEN 480
470 NEXT X
475 PRINT"BITTE CONT EINGEBEN":STOP
480 FOR Y=X+1 TO LEN(Q$)
490 IF MID$(Q$,Y,1)="/" THEN 510
500 NEXT Y
505 PRINT"BITTE CONT EINGEBEN":STOP
510 K=VAL(MID$(Q$,X+2,Y-X-2))
520 RETURN
530 DATA "4","/QSCHWIMMT ES/J2/N3/","/AFISCH","/AVOGEL"
600 PRINT:PRINT:PRINT"TIERE, DIE ICH SCHON KENNE : ":PRINT:PRINT
610 FOR I=1 TO 200
620 IF LEFT$(A$(I),2)<>"/A"   THEN 650
630 FOR Z=3 TO LEN(A$(I))
640 IF MID$(A$(I),Z,1)<>"/" THEN PRINT MID$(A$(I),Z,1);:NEXT Z
645 PRINT
646 IF I/15=INT(I/15) THEN PRINT:PRINT"BITTE CONT":STOP
650 NEXT I
660 PRINT:PRINT
700 PRINT:PRINT:PRINT:PRINT
705 IF G=1 THEN PRINT"ICH HABE RICHTIG GERATEN !!!"
706 PRINT:PRINT:PRINT
710 PRINT"NOCH EIN SPIEL (J/N) ";:INPUT A$
720 IF A$="J" THEN G=0:GOTO 120
999 PRINT:PRINT:PRINT"ENDE DER AUSGABE":END
```

13.6 Variablenliste

A$	=	Stringfeld zur Aufnahme von "J" oder "N" für "ja" bzw. "nein"
A$()	=	Tiernamen und Differenzierungsfragen
B$	=	Hilfsfeld zur Aufnahme von "J" bzw. "N" zur Koppelung mit dem Feld, in dem die jeweilige Frage steht
C$	=	wie A$; im Unterprogramm 390
G	=	Markiervariable, an der erkannt wird, ob das Programm richtig geraten hat
I	=	Laufindex
K	=	Laufindex (Kennzeichnung der Frage-Runden)
N	=	Führende Ziffern im Stringfeld A$()
Q$	=	Hilfsfeld zur Zwischenspeicherung von A$(K)
T$	=	Koppelung von "/" und C$ (s.o.)
V$	=	Name des Tiers, an das der Benutzer gedacht hat
X	=	Laufindex über die Symbole von A$()
X$	=	Frage, mit der der Benutzer differenziert
Y	=	wie X (mit anderem Startpunkt)
Z	=	wie Y
Z1	=	Führende Ziffern von A$(0)

13.7 Programmbeschreibung

Satz 10-60 :

Überschrift und Erläuterungen

Satz 70 :

Dimensionierung : Es ist Platz vorgesehen für maximal 100 Tiere und 99 differenzierende Fragen

Satz 80 :

Einlesen der anfangs zur Verfügung stehenden Hintergrundinformationen (siehe auch Satz 530)

Satz 90 :

Das Feld N wird mit den führenden Ziffern des ersten eingelesenen Strings belegt. Der Vergleich mit Satz 530 zeigt, daß dies die Zahl 4 ist

Satz 120-125 :

Der Rechner bietet dem Benutzer an, alle Tiere, die ihm schon zur Verfügung stehen, auszugeben; wenn dies gewünscht wird erfolgt ein Sprung zum Satz 600 (siehe dort)

Satz 126-150 :

Der Benutzer wird aufgefordert, an ein Tier seiner Wahl zu denken; tut er dies nicht (vielleicht deshalb, weil er nicht weiß, welche Tiere das Programm schon kennt), dann erfolgt über Satz 150 ein Rücksprung zum Satz 120

Satz 160-170 :

Mit Satz 160 beginnt die erste "Frage-Runde" (K=1). Diese Fragen werden im Unterprogramm 390 gestellt, weshalb über Satz 170 der entsprechende

Sprung erfolgt

Satz 180 :

Kommt das Programm aus diesem Unterprogramm zurück, dann wird, um Fehler zu vermeiden, an das Programmende gesprungen (Satz 999), falls im String A$(K) nichts vorhanden ist

Satz 190 :

Wenn der String A$(K) mit den Symbolen "/Q" beginnt, dann handelt es sich um eine sog. Differenzierungsfrage (Fragen und Tiernamen sind ja in dem gleichen A$-Array gespeichert, so daß eine derartige Unterscheidungsmöglichkeit erforderlich ist); in diesem Fall wird deshalb zurückgesprungen zu Satz 170, d.h. es kann die nächste Frage gestellt werden

Satz 195-200 :

Das Programm bietet nun, nachdem es per Unterprogramm 390 eine differenzierende Frage vorweggeschickt hat, ein Tier als Rateergebnis an; dabei müssen die führenden zwei Symbole der Kennzeichnung aus dem jeweiligen String weggelassen werden; deshalb die etwas umständliche Programmierung RIGHT$(A$(K),LEN(A$(K))-2)

Satz 210-220 :

Der Benutzer entscheidet jetzt, ob der Rechner richtig geraten hat oder nicht; war die Tier-Ausgabe richtig, so erfolgt ein Sprung zum Satz 700, d.h. das laufende Spiel ist (mit einem Erfolg des Rechners) beendet;

damit das Programm diesen Erfolg erkennen kann, wird vor diesem Sprung die Markiervariable G mit dem Wert 1 belegt

Satz 238-240 :

Satz 238 wird erreicht, wenn der Rechner mit

keiner der Frage-Runden (Unterprogramm 390) in der Lage war, das vom Benutzer gedachte Tier zu erraten;

in diesem Fall fordert er zunächst zur Ergänzung seiner Hintergrundinformationen den Namen des gedachten Tiers an (240)

Satz 245-270 :

Danach fordert der Rechner eine differenzierende Frage an, die das vom Benutzer ausgedachte Tier vom zuletzt genannten Tier unterscheiden läßt

Satz 274-300 :

Schließlich bittet der Rechner den Benutzer, diese differenzierende Frage für den Fall des von ihm gedachten Tieres mit "Ja" oder "Nein" (zutreffend) zu beantworten (in Satz 300 wird ein eventueller Eingabefehler abgefangen)

Satz 310-320 :

Antwortet der Benutzer auf diese Frage mit "J" (für "Ja"), so wird das Hilfsfeld B$ mit "N" belegt und umgekehrt; dadurch ist das Programm in folgenden Fragerunden in der Lage, zu erkennen, für welches Tier diese spezielle Frage mit "Ja" bzw. mit "Nein" beantwortet werden muß und kann damit dann seine "Tier-Rateversuche" steuern (vergl. zur näheren Erklärung Satz 530)

Satz 330-340 :

Nachdem sich der Informations-Datenbestand des Rechners um zwei Strings vergrößert hat (ein Tiername und eine Differenzierungsfrage), wird das Feld A$(0), welches diese Information enthält, entsprechend neu belegt

Satz 360-370 :

Entsprechend werden auch die neuen Informationen (nächster Tiername und nächste

Differenzierungsfrage) an den A$-Array angefügt :

Der Tiername in Satz 360, wobei dabei die Erkennungssymbole "/A" vorgekoppelt werden (am Symbol "/" erkennt das Programm den Anfang eines neuen Strings, am Symbol "A", daß es sich um einen Tiernamen handelt, und nicht etwa um eine Differenzierungsfrage, die mit "/Q" eingeleitet wird; siehe auch Satz 530);

die Differenzierungsfrage in Satz 370, wobei "/Q" vorgeschaltet und die J/N-Informationen (siehe Satz 310-320) mit entsprechenden Feldindexwerten angeschlossen werden (siehe zur Erklärung auch Satz 530)

Satz 380 :

Rücksprung zu Satz 120, d.h. Beginn eines neuen Spiels, wenn der Benutzer dies wünscht

Satz 390-395 :

Beginn des Unterprogramms, das die Differenzierungsfragen stellt

Satz 400 :

Das Hilfsfeld Q$ wird mit dem jeweiligen String A$(K) belegt; wenn dieses Unterprogramm zum ersten Mal angesprungen wird, ist dies A$(1), weil K=1; dies bedeutet, daß d.h. die erste Differenzierungsfrage gestellt wird (siehe auch Satz 530)

Satz 410-415 :

Dieser String, also diese Differenzierungsfrage wird beim dritten Symbol beginnend (um die Symbole "/Q" zu unterdrücken) und vor dem nächsten Schrägstrich endend, ausgegeben

Satz 420-440 :

Der Benutzer wird aufgefordert, diese Frage mit

"J" oder "N" (für "Ja" oder "Nein") zu beantworten; dabei wird wieder eine eventuelle fehlerhafte Eingabe abgefangen (440)

Satz 450 :

Im Feld T$ wird die Antwort des Benutzers, mit einem Schrägstrich gekoppelt, gespeichert (vergl. wieder Satz 530)

Satz 455-470 :

Es wird untersucht, wo im String Q$, beginnend beim dritten und endend beim vorletzten Symbol, die Symbolfolge "/J" oder "/N" vorkommt; ist diese Stelle gefunden, wird die Schleife mit einem Sprung zum Satz 480 verlassen, wobei X die Symbolposition angibt, bei der dieses Verlassen erfolgte

Satz 475 :

Ist dies nicht der Fall, so wird das Programm unterbrochen und kann nach Eingabe des Kommandos CONT fortgesetzt werden

Satz 480-500 :

Satz 480 wird erreicht, wenn die Symbolfolge T$ aufgetaucht ist (siehe Satz 455-470); zusätzlich wird nun, ab derjenigen Symbolposition X, bei der die vorhergehende Schleife verlassen wurde (455-470), untersucht, wo das Symbol "/" auftaucht;

ist dies gefunden, so erfolgt ein Sprung zum Satz 510

Satz 505 :

Dieser Satz bewirkt wieder eine Unterbrechung des Programms; nach Eingabe des Kommandos CONT wird das Programm fortgesetzt

Satz 510 :

Wenn im String Q$ ein Schrägstrich gefunden wurde, wird K neu belegt, indem aus dem String Q$ (also aus der Differenzierungsfrage) die Ziffer, die bei J, oder die Ziffer, die bei N steht (siehe Satz 530 als Beispiel), herausgegriffen wird, je nachdem, ob der Benutzer auf die Frage zur Differenzierung mit "J" oder "N" geantwortet hatte (dies schlug sich ja in T$ nieder (siehe 450);

dies wiederum bestimmte die Festlegung von X, mit der die Schleife 455-470 verlassen wurde und dies wiederum bestimmte dann auch Y, mit der die Schleife 480-500 verlassen wurde)

Satz 520 :

Rücksprung (mit neuem K-Wert) ins rufende Hauptprogramm

Satz 530 :

Erste Informationsbasis des Programms :

Der A$-Array, der mit diesen Informationen belegt wird, umfaßt also zunächst vier Felder mit den folgenden Belegungen :

1. Feld :

Hier steht die Zahl 4, d.h. es wird angegeben, daß die Ausgangsinformationen aus jeweils vier Teilen bestehen; dieses Feld wird in Satz 340 in folgenden Fragerunden neu belegt

2. Feld :

Hier steht die erste Differenzierungsfrage; sie wird mit den Symbolen "/Q" eingeleitet und mit "/J2/N3" beendet;

über die beiden ersten Symbole wurde oben schon gesprochen; die angehängte Symbolkette erlaubt dem Programm, zu erkennen, daß es mit "Fisch"

reagieren muß, wenn der Benutzer auf die Frage "Schwimmt es ?" mit "J" geantwortet hat, hingegen mit "Vogel", wenn er mit "N" geantwortet hat;

es ist natürlich wichtig, daß bei neu aufgenommenen Differenzierungsfragen diese Teile korrekt belegt werden, damit diese neuen Fragen in Verbindung mit den neu aufgenommenen Tiernamen korrekte Entscheidungen des Programms ermöglichen (siehe dazu den etwas komplizierten Satz 370)

Feld 3 :

Hier findet sich der erste Tiername (eingeleitet, wie schon erläutert, mit den Symbolen "/A")

Feld 4 :

Zweiter Tiername

Satz 600-650 :

Ausgabe der Tiernamen, die das Programm schon kennt, wenn der Benutzer dies wünscht (Satz 600 wird nur über einen Sprung von Satz 125 ausgehend erreicht)

Bei dieser Ausgabe muß geprüft werden, ob die Inhalte des A$-Arrays mit "/A" anfangen (Satz 620), denn nur dann handelt es sich um Tiernamen (die Differenzierungsfragen, die im gleichen Array stehen, sollen ja nicht ausgegeben werden);

die Tiernamen selbst werden Symbol für Symbol ausgegeben (Satz 630-640), wenn das jeweilige Symbol kein Schrägstrich ist (640);

nach je 15 Tiernamen wird die Ausgabe unterbrochen (Fortsetzung mit dem Kommando CONT), damit der Programmbenutzer Gelegenheit hat, die Tiernamen in Ruhe zu lesen (646)

Satz 660-999 :

Nach Ausgabe einiger Leerzeilen, wird der Programmbenutzer gefragt, ob er ein weiteres Spiel wünscht;

ist dies der Fall, so erfolgt ein Rücksprung zum Satz 120, nachdem die Markiervariable G wieder auf null zurückgesetzt wurde; andernfalls wird mit Satz 999 das Programm beendet

13.8 Programmergebnisse

Wenn wir dieses Programm mit dem Kommando RUN starten, so fordert es uns nach der Ausgabe einer Überschrift auf, uns ein bestimmtes Tier auszudenken und fragt danach, ob es vor dem nächsten Spiel einmal zunächst alle Tiere ausgeben soll, die es schon kennt.

Wenn wir diese Frage mit "J" (für "Ja") beantworten, so produziert das Programm auf dem Bildschirm die folgende Ausgabe :

```
TIERE, DIE ICH SCHON KENNE :

FISCH       VOGEL


NOCH EIN SPIEL (J/N) ?
```

Antworten wir jetzt wieder mit "J", so fragt das Programm erneut, ob wir zunächst seine schon vorhandenen Kenntnisse erfahren wollen.

Wenn wir jetzt mit "N" (für "Nein") antworten, so werden wir gefragt, ob wir jetzt an ein Tier denken.

Nehmen wir einmal an, wir denken an "Maus" und antworten deshalb auf diese Frage mit "J". Dann fragt das Programm, ob dieses Tier schwimmt (das ist die erste und bislang einzige Frage zur Differenzierung, die dem Programm zur Verfügung steht).

Wenn wir diese Frage verneinen, dann "rät" das Programm :

HEISST DAS TIER VOGEL (J/N) ?

Auch diese Frage müssen wir verneinen und deshalb verlangt nun das Programm (zum Selbstlernen) den Namen des Tieres, das wir uns gedacht haben :

Wahrheitsgemäß geben wir also MAUS ein und das Programm antwortet daraufhin :

BITTE GIB EINE FRAGE EIN, BEI DEREN
BEANTWORTUNG MIT J (FUER JA) ODER N
(FUER NEIN) SICH

MAUS UND VOGEL

VONEINANDER UNTERSCHEIDEN.

BEISPIEL : KANN DAS TIER BEISSEN

FRAGE : ?

Wir könnten also jetzt zum Beispiel die folgende Frage formulieren :

KANN DAS TIER FLIEGEN

Wieder soll nun mit ja oder nein geantwortet werden. Da es sich um eine Maus handelt, antworten wir auf die Frage, ob das Tier fliegen kann mit "N".

Damit ist die erste Runde beendet. Das Programm hat das gedachte Tier (Maus) nicht erraten können, hat aber seinen Informationshintergrund um ein drittes Tier (Maus) und um eine zweite

Differenzierungsfrage (kann das Tier fliegen ?) erweitern können.

Dem Programmbenutzer bleibt es überlassen, zu entscheiden, ob nun ein nächster Programmlauf beginnen soll.

Wenn ja, kann beispielsweise wieder zuerst ausgegeben werden, welche Tiere das Programm schon kennt. Es antwortet dann :

TIERE, DIE ICH SCHON KENNE :

FISCH VOGEL MAUS

Der Programmlauf kann dann so fortgesetzt werden, wie es oben schon beschrieben wurde.

13.9 Ausblick

Die Tatsache, daß wir hier auf der Grundlage eines spielerischen Beispiels die Möglichkeiten selbstlernender Programme illustriert haben, darf nicht darüber hinwegtäuschen, daß im praktischen Einsatz derartig strukturierte Programme eine zunehmend wichtiger werdende Rolle spielen.

Insbesondere im Zusammenhang mit Experten- und Auskunftssystemen sind die Fähigkeiten der selbsttätigen Wissenserweiterung von Programmen von außerordentlich großer Bedeutung, wobei zusätzlich ein positiver Effekt der Beschleunigung zutage tritt :

Je öfter ein derartiges System benutzt wird, desto mehr Wissen akkumuliert es und desto größer werden die Chancen, mit dem Programmsystem zu korrekten Entscheidungen zu gelangen. Die praktische Nutzung derartiger Systeme schreitet deshalb in fast allen

Bereichen rasch voran.

Das vorangegangene Beispiel eignet sich sehr gut auch dazu, auf ein besonders schwieriges Problem derartiger selbstlernender Programme aufmerksam zu machen :

Der "Lernerfolg" eines derartigen Systems hängt in entscheidender Weise davon ab, ob wir - um bei unserem Beispiel zu bleiben - dem Progamm korrekte Differenzierungsfragen vorgeben.

Als wir uns in unserem Beispiel das Tier "Maus" ausdachten, die das Programm im ersten Durchgang nicht erraten konnte, wurden wir aufgefordert, den Tiernamen und eine Frage einzugeben, die dieses Tier (Maus) vom vorher genannten Tier (Vogel) untercheiden kann und die Antwort auf diese Frage.

Deshalb hatten wir als Frage

"KANN DAS TIER FLIEGEN"

eingegeben und als Antwort mit Hinblick auf die Maus natürlich "Nein".

Gesetzt den Fall nun aber, die Frage differenziert nicht hinreichend. Schon bei der ersten Frage ("schwimmt es ?") ist dies eigentlich der Fall. Diese Frage soll ja eine Entscheidung zwischen "Fisch" und "Vogel" ermöglichen. Jeder weiß aber, daß es auch Vögel gibt, die schwimmen können (die Ente zum Beispiel) und selbst von fliegenden Fischen hat man schon gehört.

Dies bedeutet, daß trotz der vorhandenen Hintergrundinformationen und vorhandener Differenzierungsfragen möglicherweise Fehlentscheidungen zustandekommen können, wenn diese Fragen nicht völlig sauber zwischen zwei Tierarten trennen können.

Worauf es in diesem Zusammenhang also ankommt, ist die erfolgreiche Suche nach völlig sauber trennenden Differenzierungsfragen, sonst kann ein

selbstlernendes Programmsystem seine Fähigkeiten nie voll entfalten.

Dieser Problembereich wird in der Fachliteratur unter dem neuen Stichwort "Knowledge Engineering" ausführlich diskutiert, so daß wir an dieser Stelle den interessierten Leser darauf verweisen dürfen (siehe z.B. RETTI,J. : 1984).

Kapitel 14 : Kunst

14.1 Aufgabenstellung

Im ersten Teil dieses Buches wurde schon darauf hingewiesen, daß die Frage, ob Computer wirklich intelligent seien oder eines Tages sein könnten, häufig deshalb verneint wird, weil man feststellt, Computer könnten keine Gedichte schreiben und keine Streichquartette komponieren.

Wir wollen in diesem Kapitel ansatzweise zeigen, daß Computer auch vor derartigen Aufgaben nicht zurückschrecken, wobei aber einige einschränkende Vorbemerkungen angebracht sind :

Ein Gedicht als Ergebnis künstlerischen Schaffens geht immer aus mehreren Antriebskräften hervor. Da ist einmal die inhaltliche Aussage, das also, was der Künstler mit seinem Gedicht mitteilen möchte; da ist weiterhin die emotionale Komponente, d.h. Gefühle bestimmen die Entstehung und die Aussage des Gedichts in weitem Maße, Gefühle sollen beim Leser oder Zuhörer auch geweckt werden; und schließlich geht es um die Form, d.h. das Kunstwerk soll bestimmten formalen Kriterien genügen.

Was für ein Gedicht gilt, gilt in entsprechender Weise auch für andere Bereiche des künstlerischen Schaffens, also zum Beispiel auch für die bildende Kunst oder für die Musik.

Es ist unmittelbar einsichtig, daß insbesondere der emotionale Aspekt, aber sicher auch der der inhaltlichen Aussagen nur schwer oder vielleicht gar nicht durch Computerprogramme nachbildbar ist. Wenn man aber zunächst einmal - was sicher eine

sehr einschränkende Betrachtungsweise ist - die rein formalen Aspekte dominieren läßt, dann ergeben sich auch für Computer interessante Einsatzgebiete :

Es gab und gibt Kunstrichtungen - etwa in der Malerei - bei denen zunächst einmal das freie Hantieren mit Farben und Formen im Vordergrund steht, ein eher formaler Aspekt also, der die eventuellen Inhalte deutlich in den Hintergrund rückt, wenn auch emotionale Gesichtspunkte dabei eine Rolle spielen : Die Zusammenstellung unterschiedlich abgestufter Blautöne zu einem teuren Kunstobjekt ist ja nicht nur unter formalen Gesichtspunkten interessant, sondern weist auch eine nicht übersehbare emotionale Komponente auf.

Ähnlich geht man von Zeit zu Zeit in der modernen Lyrik vor, wo man beispielsweise aus einem gegebenen Wortvorrat durch zufallsbeeinflußte Mischungen und Zusammenstellungen unter Beachtung einschränkender formaler Kriterien (z.B. soll sich nicht das Wort "und" direkt wiederholen können oder dergl.) interessante und sogar unter den emotionalen Aspekten interpretierbare Gedichte erzeugen kann.

Ähnliches gilt schließlich im Bereich der Musik : Wenn man einen Computer mit grundlegenden Kompositionsregeln versorgt und ihm dann als "Spielmaterial" einen bestimmten Tonvorrat vorgibt, dann kann er Kompositionen im Stile Bachs oder Schuberts oder Bartoks erstellen - Nachahmungen gewissermaßen; insoweit könnte man das Stichwort "Kunst" in der Überschrift dieses Kapitels in Anführungszeichen setzen.

Wie im einzelnen bei derartigen "Nachahmungen" vorgegangen werden kann, soll anhand eines einfachen Beispiels in diesem Kapitel gezeigt werden.

14.2 Lösungsansätze

Wenn man in dem oben beschriebenen Sinne einen Computer veranlassen will, ausgehend von einem eingegebenen Programm, "Kunst" zu erzeugen, dann ist der prinzipielle Lösungsweg schon deutlich vorgezeichnet :

Der erste Schritt muß darin bestehen, den Rechner mit "Spiel-Bausteinen" zu versehen : Tonvorgaben, Worte, geometrische Figuren, Buchstaben, Ziffern, Farben usw., all dies können solche Bausteine sein.

Damit er nun mit derartigen Bausteinen etwas anfangen kann, benötigt der Rechner Auswahlregeln (welche der Bausteine sollen verwendet werden, welcher zuerst usw.) und Kompositionsregeln (in welcher Art und Weise sind die Bausteine dann zusammenzufügen, welche Kombinationen sind erlaubt, gewünscht, welche sollen ausgeschlossen sein usw.).

Bausteine und Regeln zusammen erlauben es dem Rechner "künstlerische" Ergebnisse zu produzieren, die er uns dann zum Beispiel auf dem Bildschirm präsentieren kann.

Da der Rechner in der Lage ist, uns in sehr kurzer Zeit sehr viele Variationen zu bieten, können wir per Programm sehr vieles durchprobieren lassen, um uns dann diejenigen Ergebnisse herauszufischen, die uns am besten gefallen, und die vielleicht auch der etablierten Kunstkritik entsprechen können.

14.3 Das Beispiel

Wie auch schon in den vorangegangenen Kapiteln steht auch hier der Gedanke im Mittelpunkt, das Demonstrationsbeispiel so anzulegen, daß der Leser ohne große Schwierigkeiten in der Lage ist, die Grundprinzipien eines derartigen Kunstprogramms zu erkennen. Deshalb gehen wir von dem folgenden Beispiel aus :

Das Programm, um das es hier geht, erzeugt ein farbiges geometrisches Muster. Es zeichnet eine Reihe verschiedenfarbiger Rechtecke auf dem Bildschirm (vorausgesetzt man hat dabei einen Farbfernseher angeschlossen), wobei die Anzahl dieser Rechtecke vom Programmbenutzer vorgegeben werden kann.

Die Lage, die Größe und die Farbe dieser Rechtecke werden per Zufallsprinzip bestimmt, so daß sich bei jedem Programmlauf ein anderes Bild ergeben kann. Das schönste Bild kann dann der Benutzer des Programms als Computerkunstwerk per Farbphoto oder über einen farbfähigen Drucker für zukünftige Generationen aufbewahren.

14.4 Problemanalyse

Die Problemanalyse ist bei dieser Aufgabenstellung sehr einfach :

Der Rechner benötigt als Inputinformation die Zahl der gewünschten Rechtecke, bestimmt für jedes Rechteck mit Hilfe der RND - Funktion einen Zufallspunkt auf dem Bildschirm, der als der obere, linke Eckpunkt des Rechtecks definiert wird, bestimmt weiterhin für jedes Recheck nach dem Zufallsprinzip Länge und Breite, bestimmt schließlich, wiederum nach dem Zufallsprinzip eine der zur Verfügung stehenden Farben, und kann danach Rechteck für Rechteck zeichnen.

Es ist bei dieser Vorgehensweise lediglich darauf zu achten, daß beim Zeichnen der zulässige Bildschirmspeicherbereich von der Adresse A = 1024 (links oben) bis zur Adresse A = 2023 (rechts unten) nicht überschritten wird.

Zur Einhaltung dieser Bedingung gibt es mehrere Wege, beispielsweise die folgenden beiden :

1. Jedesmal, bevor eine Bildschirmspeicheradresse belegt werden soll, wird geprüft, ob sie noch zulässig ist. Wenn dies nicht der Fall ist, wird nicht gezeichnet, sondern die nächste Adresse in Angriff genommen.

2. Man kann die Zufallsauswahl der linken oberen Ecke des Rechtecks und die Zufallsauswahl von Länge und Breite des jeweiligen Rechtecks so beschränken, daß Eckpunkt + Länge nicht über den linken Bildschirmrand und Eckpunkt + Breite nicht über den unteren Bildschirmrand hinauswandern kann.

Im Programm haben wir den zweiten Weg gewählt, wobei man sich daran erinnern muß, daß man sich den Bildschirm in 40 Spalten (0 bis 39) und 25

Zeilen (0 bis 24) aufgeteilt denken kann (dem entsprechen die 1000 Bildchirmspeicheradressen von 1024 bis 2023).

Jedes Rechteck, das gezeichnet werden soll, bauen wir aus kleinen Quadraten auf, die jeweils eine Bildschirmspeicheradresse A belegen. Ein solches Quadrätchen wird also mit der Anweisung

```
POKE A,224
```

gezeichnet, wobei 224 die Bildschirmcodezahl für ein inverses leeres Quadrätchen ist (es muß die inverse Darstellung gewählt werden, weil das leere Quadrat blau auf blauem Hintergrund nicht sichtbar ist; in der inversen Darstellung wird es bei gegebener Farb-Voreinstellung hellblau auf blau).

Die Einfärbung dieses Quadrats wird dadurch bewerkstelligt, daß wir die jeweils zugehörige Farbspeicheradresse A + 54272 dann mit einer Farbcodezahl F (0 bis 15) belegen, also :

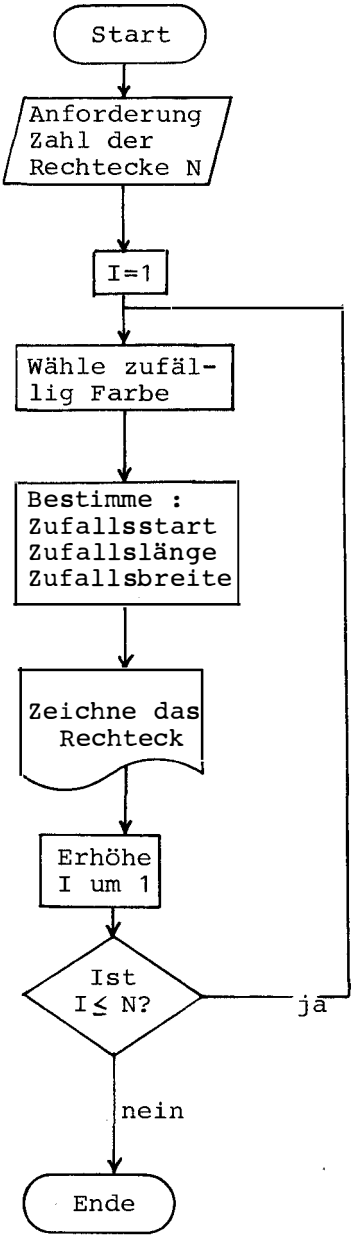
```
POKE A+54272,F
```

Dabei sollte die Farbe F = 6 übersprungen werden, denn dies ist blau : Ein so gezeichnetes Rechteck bliebe auf dem blauen Hintergrund unsichtbar.

Wenn man die obigen POKE-Anweisungen nun in Schleifen einbaut, dann werden die einzelnen Quadrätchen zu Balken und die Balken zu Flächen zusammengefügt und es ensteht ein Zufallsbild auf dem Farbmonitor.

Gemäß diesen Überlegungen gelangen wir zu dem folgenden Programmablaufplan :

Flußdiagramm "Kunst"



14.5 Programm

```
10 REM KI08 - KUNST
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,     1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 8 :";
60 PRINT "       KUNST"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM DIENT DAZU, EIN GEOME-"
90 PRINT"TRISCHES MUSTER ZU ERZEUGEN, DAS BEI "
100 PRINT"JEDEM PROGRAMMLAUF ANDERS AUSFAELLT."
110 PRINT:PRINT:PRINT:PRINT
120 PRINT"ES WERDEN EINE BELIEBIGE ANZAHL FARBI-"
130 PRINT"GER RECHTECKE UNTERSCHIEDLICHER GROESSE"
140 PRINTTAB(16)"ERZEUGT.":PRINT:PRINT:PRINT
150 INPUT "WIEVIELE RECHTECKE WERDEN GEWUENSCHT";N
160 PRINTCHR$(147)
170 FOR I=1 TO N
180 F=INT(RND(1)*16):IF F=6 THEN F=F+1
190 S=INT(RND(1)*25):Z=INT(RND(1)*15)
200 L=INT(RND(1)*14+1):B=INT(RND(1)*8+1)
210 SA=1024+Z*40+S
220 FOR J=SA TO SA+B*40 STEP 40
230 FOR K=J TO J+L
240 IF K<1024 OR K>2023 THEN 260
250 POKE K+54272,F:POKE K,224
260 NEXT K
270 NEXT J
280 NEXT I
290 END
2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
```

14.6 Variablenliste

A$	=	Stringvariable zur Tastatureingabe
B	=	Breite des Rechtecks (in Einheiten von Bildschirmadressen)
F	=	Farbcodezahl (0 bis 15)
I	=	Laufindex (über die Rechtecke)
J	=	Laufindex (über die Zeilen im Rechteck)
K	=	Laufindex (über die Spalten im Rechteck)
L	=	Länge des Rechtecks (siehe B)
N	=	Zahl der Rechtecke
S	=	Startspalte
SA	=	Startadresse
Z	=	Startzeile

14.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tasteneingabe

Satz 80-140 :

Erläuterungen zum Programm

Satz 150-160 :

Anforderung der Zahl der gewünschten Rechtecke und Löschen des Bildschirms vor der graphischen Ausgabe

Satz 170 :

Beginn der Schleife "über alle Rechtecke"

Satz 180 :

Bestimmung einer zufälligen Zeichenfarbe;

wenn der Farbcode F=6 ist (blau), wird F um 1 erhöht, damit nicht blau auf blauem Hintergrund, also unsichtbar gezeichnet wird

Satz 190 :

Zufällige Bestimmung der Bildschirmspalte der linken oberen Ecke des Rechtecks (es wird dabei nicht die gesamte Bildschirmbreite genutzt, sondern der Maximalwert ist - willkürlich - auf 24 festgesetzt); danach entsprechende Zeilenbestimmung (Maximalwert 14)

Satz 200 :

Zufallsabhängige Längen- und Breitenbestimmung (Maximalwert der Länge : 14, Minimalwert : 1 - nicht etwa 0, was durch die Addition des Wertes 1 in der INT-Funktion erreicht wird; der Leser überlege, warum wir mit dem eventuellen Wert 0 nichts anfangen könnten; Maximalwert der Breite : 8)

Satz 210 :

Bestimmung der Startadresse : Wenn Spalte und Zeile festgelegt sind (S und Z), so erhält man die zugehörige Bildschirmspeicheradresse, indem man zum Anfangswert 1024 (links oben) die Zahl der Spalten (S) und die Zahl der Zeilen (Z), multipliziert mit 40 (weil in jeder Bildschirmzeile 40 Adressen sind) hinzuaddiert

Satz 220-270 :

Schleife zum Zeichnen der Zeilen innerhalb eines Rechtecks; deshalb Schrittweite 40 in Satz 220;

diese Schleife beginnt bei der Startadresse SA und endet bei Startadresse + Breite des Rechtecks * 40, weil ja jede Folgezeile Adressen aufweist, die um 40 höher liegen als die der vorhergehenden Zeile;

innerhalb dieser Zeilenschleife wird in Satz 230-260 Spalte für Spalte ein Quadrätchen in einer bestimmten Farbe gezeichnet (250);

diese Spaltenschleife beginnt beim jeweiligen Zeilenindex J und endet bei J + Länge des Rechtecks, also bei J+L;

der Sicherheit halber wird dabei jeweils in Satz 260 noch einmal überprüft, ob auch wirklich nicht der zulässige Bereich von Bildschirmbereich verlassen wird; sollte dies einmal der Fall sein, wird die Zeichenanweisung übersprungen

Satz 280 :

Inangriffnahme des nächsten Rechtecks

Satz 290 :

Beendigung des Programms

Satz 2000-2040 :

Unterprogramm zum Abwarten einer Tastatureingabe

14.8 Programmergebnisse

Auf die Darstellung von Programmergebnissen kann hier verzichtet werden. Sinnvoller und auf jeden Fall anschaulicher ist es, wenn der Benutzer das Programm einfach einmal ausprobiert.

14.9 Ausblick

Wir haben mit diesem Programm ein Beispiel aus einem ganz anderen Bereich vorgestellt. Der Leser kann sich sicherlich nach dem Studium dieses Programms und der Programmbeschreibung leicht vorstellen, in welcher Weise ein solches Programm verändert, ergänzt oder verbessert werden kann :

Beispielsweise könnte es sinnvoll erscheinen, nur bestimmte Farbkombinationen zuzulassen oder andere geometrische Figuren zu verwenden oder sog. Bewegungselemente in das Programm mit aufzunehmen (vergl. dazu das übernächste Kapitel) usw.

Der Phantasie des Benutzers sind keine Grenzen gesetzt. Er erprobe beispielsweise einmal das folgende kleine Programm :

```
10 REM FARBSPIEL
20 PRINTCHR$(147)
30 FOR A=1024 TO 2023
40 F=INT(RND(1)*14+1)
50 POKE A+54272,F:POKE A,224
60 NEXT A
70 END
```

Mit einem derartigen Programm kann man einfache graphische und farbliche Muster erzeugen, die man aber auch beliebig verkomplizieren kann.

Nicht zuletzt deshalb wird den Möglichkeiten der Computergraphik in letzter Zeit immer mehr Aufmerksamkeit gewidmet. Vor allem beim Einsatz leistungsfähiger Großrechner ist die Herstellung hervorragender und teilweise auch durchaus sehr ansprechender Graphiken möglich geworden, die sich insbesondere auch durch ihre Bewegungskomponenten auszeichnen.

Die Anfertigung von Werbegraphiken, von Plakaten, von Werbefilmen u. dergl. wird heutzutage von Computern und ihren Programmen übernommen, die den traditionellen Werbegraphiker, der mit Tusche, Farben und Spritzpistolen hantierte, überflüssig machen.

Dies geht schließlich so weit, daß die alte Kunst

der Herstellung von Zeichentrickfilmen in Vergessenheit gerät, weil inzwischen etwa Bewegungsabläufe in hoher zeichnerischer und farblicher Qualität von Rechnern erzeugt werden können.

Hier eröffnet sich ein zunehmend breiter werdendes Anwendungsgebiet, das uns die Gelegenheit gibt, auch einmal auf zukünftig wichtig werdende berufliche Möglichkeiten und Veränderungen traditioneller Arbeitsplätze aufmerksam zu machen.

Kapitel 15 : Dialog

15.1 Aufgabenstellung

Nach unserem Ausflug in die Kunst wollen wir uns einem anderen Bereich zuwenden, der unter dem Stichwort "künstliche Intelligenz" ebenfalls häufig genannt wird : den Bereich der Gespräche und Dialoge zwischen Mensch und Computer.

Berühmt geworden ist das Beispiel des Programms ELIZA, welches den Psychotherapeuten in seinem therapeutischen Gespräch mit einem Patienten ersetzen konnte - oder, wie die engagierten Kritiker dieses Programms einwandten, eben nicht ersetzen konnte.

Bei diesem Programm ELIZA ging man, wie bei allen anderen Gesprächsprogrammen auch, zunächst von der Vorstellung aus, daß sich intelligentes Verhalten eines Computers darin dokumentiert, daß er über ein bestimmtes Thema mit dem Computerbenutzer ein vernünftiges Gespräch führen kann.

Wenn ein solches Programm so gut ist, daß der menschliche Gesprächspartner nicht erkennen kann, daß er mit einer Maschine kommuniziert, also keinen Unterschied zum Beispiel zwischen einem zuhörenden, fragenden und antwortenden Therapeuten und einer Maschine, die ebenfalls zuhört, fragt und antwortet, erkennen kann, dann hat er es - so darf man definieren - mit einem intelligenten Programm, bzw. mit einer intelligenten Maschine zu tun.

Der Versuch, mit einer derart programmierten Maschine den Psychotherapeuten, den Arzt, den Seelsorger oder den liebevollen Ehepartner zu

ersetzen (zumindest soweit Gespräche betroffen sind), mag vermessen erscheinen und aus den Ausführungen des ersten Teils dieses Buches dürfte deutlich geworden sein, daß hier sicherlich nicht allzu viele Erfolge erwartet werden dürfen - gleichwohl wäre ein dialogfähiger Rechner sicherlich eine sinnvolle Unterstützung zum Beispiel im Rahmen von Experten- oder von Auskunftssystemen.

Das oben erwähnte Programm ELIZA stammt übrigens von dem berühmten J.WEIZENBAUM, der zum sehr engagierten Kritiker des häufig zu unüberlegten Computereinsatzes geworden war, als er erkannte, welch vermessene Hoffnungen man auf der Grundlage von ELIZA in zukünftige Dialogprogramme setzte :

Manche Psychotherapeuten gingen nämlich ab und zu tatsächlich so weit, Routinegespräche mit Patienten an derart programmierte Rechner zu delegieren, ohne zu bedenken, daß ein Gespräch von Mensch zu Mensch doch auch Elemente enthält, die beim Informationsaustausch mit einem Computer nicht auftauchen können, gleichwohl aber - vor allem im psychotherapeutischen Bereich - außerordentlich wichtig sind.

Einem Computer das Sprechen beizubringen, so wie dem Mädchen Eliza in dem Stück "Pygmalion" von G.B.SHAW, ist nicht sonderlich schwer; ihn aber mit all denjenigen Charakteristika auszustatten, die ein Gespräch von Mensch zu Mensch immer mitbestimmen, ist sicherlich ein unlösbares Problem.

Deshalb müssen wir uns im folgenden auf eher schematische Gespräche beschränken : Man denke beispielsweise an einen Computer, der Zugauskünfte im Dialog mit dem Fahrgast bereitstellt, an Computer, die automatisch Hotelreservierungen im Gespräch mit dem Reisenden veranlassen, an solche Informationssysteme, die im Dialog die von dem Benutzer gewünschten Auskünfte bereitstellen usw.

Bei diesen und ähnlichen Problemstellungen ist die

Programmierung nicht so schwer, wie man sich vielleicht zunächst vorstellt.

15.2 Lösungsansätze

Wenn ein Rechner ein Gespräch mit dem Benutzer führen will, dann benötigt er entsprechende Hintergrundinformationen. Dieses Wissen setzt sich in der Regel aus unterschiedlichen Komponenten zusammen, deren wichtigsten die folgenden sind :

1. Der Rechner muß über Sachinformationen verfügen, wenn er Auskünfte geben soll : Der Fahrplancomputer muß die Abfahrtszeiten der Züge kennen, Zielorte und Zwischenstationen, die Umsteigemöglichkeiten oder -notwendigkeiten, die Wagenklassen, Fahrpreise usw.

2. Der Rechner benötigt einen Informationsvorrat, der es ihm erlaubt, die ihm gestellten Fragen des Gesprächspartners zu erkennen, um auf diese dann angemessen reagieren zu können.

3. Gegebenenfalls müssen Regeln vorhanden sein, die ihn zum Beispiel erkennen lassen, welche Fragen sinnvoll und zulässig sind und welche nicht ("wann fährt ein Zug am 31.Februar, nach 27 Uhr nach Köln ?" ist sicherlich keine sinnvolle Frage).

4. Schließlich muß, wenn nicht nur ein simpler Auskunftscomputer entstehen soll, sondern "echte" Gespräche zustandekommen sollen, der Rechner auf Aussagen des menschlichen Gesprächspartners eingehen können.

Dieser zuletzt genannte Punkt beinhaltet bei der Entwicklung der entsprechenden Programme die hauptsächlichen Schwierigkeiten.

In einfacheren Programmversionen versucht man,

diese Schwierigkeiten so zu lösen, daß das Programm sich aus der Aussage des menschlichen Gesprächspartners bestimmte Schlüsselworte herausgreift und diese dann seinerseits in der Gegenrede verwendet.

Die sicherlich einfachste Form einer derartigen Reaktion besteht darin, die Aussage des Menschen schlicht als Frage zu wiederholen, wie das folgende Beispiel zeigt :

Mensch : "Meine Frau versteht mich nicht"

Computer : "Ihre Frau versteht sie nicht ?"

Aber auch die Programmierung von Dialogen, die so einfach strukturiert sind, bietet schon erhebliche Schwierigkeiten, die insbesondere in der großen Kompliziertheit der deutschen Grammatik begründet liegen. Aber selbst im Englischen, wo doch die grammatikalischen Probleme nicht ganz so groß sind, gibt es noch Schwierigkeiten genug, wie zum Beispiel in dem Buch von BRAIN & BRAIN anschaulich dargestellt wird (BRAIN,K. /BRAIN,S. : 1984, Kap.4).

Diese Vorgehensweise, nämlich auf die Aussage des menschlichen Gesprächspartners in veränderter Form wieder einzugehen (was den Vorteil hat, unendlich lange Gespräche mit minimalen Informationen zu ermöglichen), war auch wesentliches Grundelement des Programms ELIZA, das oben erwähnt wurde.

Ein Gespräch nach diesem Muster könnte folgendermaßen aussehen :

Mensch : Ich habe ein ernstes Problem

Computer : Wie kann ich bei dem ernsten Problem helfen ?

Mensch : Es hat mit mir und meinem Lehrer zu tun

Computer : Sind Sie sicher, daß es mit Ihnen und Ihrem Lehrer zu tun hat ?

Mensch : Ja, ganz gewiß

Computer : Sie sind sich dieser Sache sehr gewiß ?

usw.

(Dies ist ein relativ frei ins Deutsche übersetzte Auszug aus einem Gespräch, wie es mit dem Programm ELIZA zustandekommen könnte; siehe dazu auch : HARTNELL,T : 1984, S. 182 ff.)

Man sieht an diesem kurzen Ausschnitt sicherlich sehr deutlich, wo die wesentlichen Probleme und damit die Lösungsansätze liegen :

- Auffinden der zentralen Begriffe in der Aussage des Menschen, d.h. Unterscheidung in wichtige und unwichtige Begriffe in dieser Aussage;

- Einbau dieser Begriffe in die Antwort des Rechners und eventuelle Ergänzung um solche Schlüsslworte, die das Gespräch weiterbringen;

- Ermittlung der korrekten grammatikalischen Struktur der Antwort des Rechners

usw.

Wir wollen auf weitere Einzelheiten hier nicht eingehen, sondern anhand eines einfachen Beispiels die zentralen Grundprinzipien vorführen.

15.3 Das Beispiel

Das folgende Programm soll ein Gespräch mit dem Programmbenutzer über die Qualitäten seines Computers zustandebringen.

Um dies zu ermöglichen, versehen wir das Programm mit notwendigen Hintergrundinformationen, nämlich zunächst einmal mit einer Gruppe von Fragen, die sich auf die Qualitäten des Computers beziehen. Diese Fragen werden dem Benutzer im Gespräch vom Rechner gestellt, und der Benutzer soll dann diese Fragen beantworten.

Um die Programmstruktur von Vornherein spürbar zu vereinfachen, haben wir die Fragen so vorgegeben, daß sie vom Benutzer nur mit "ja" oder "nein" beantwortet werden können und sollen.

Um das Gespräch nun weiterzuführen, muß dann der Rechner wiederum auf die "Ja"- bzw. "Nein" - Eingaben des Benutzers reagieren.

Die Fragen sind so angelegt, daß die Antwort "Ja" eine positive, die Antwort "Nein" hingegen eine negative Äußerung bezüglich der Qualitäten des Computers darstellt. Deshalb wird jetzt in einem dritten Schritt vorgesehen, daß das Programm auf eine "Ja"-Eingabe erfreut, auf eine "Nein"-Eingabe hingegen mit Bedauern reagiert.

Um gewisse Eintönigkeiten im Dialog zu vermeiden, befinden sich im Informationshintergrund des Rechners fünf verschiedene Antworten, die eine erfreute Reaktion dokumentieren und ebenfalls fünf verschiedene Antworten, die eine Reaktion des Bedauerns zum Ausdruck bringen.

Aus diesem Vorrat wählt das Programm per Zufall eine jeweils passende Antwort aus und erst dann wird die nächste Frage zur Qualität des Computers in Angriff genommen.

Wenn alle Fragen gestellt sind, kann das Programm schließlich anhand des Verhältnisses zwischen "Ja"- und "Nein"-Antworten des Benutzers erkennen, wie zufrieden der Benutzer insgesamt mit der Qualität des Rechners ist und eine entsprechende Bewertung ausgeben.

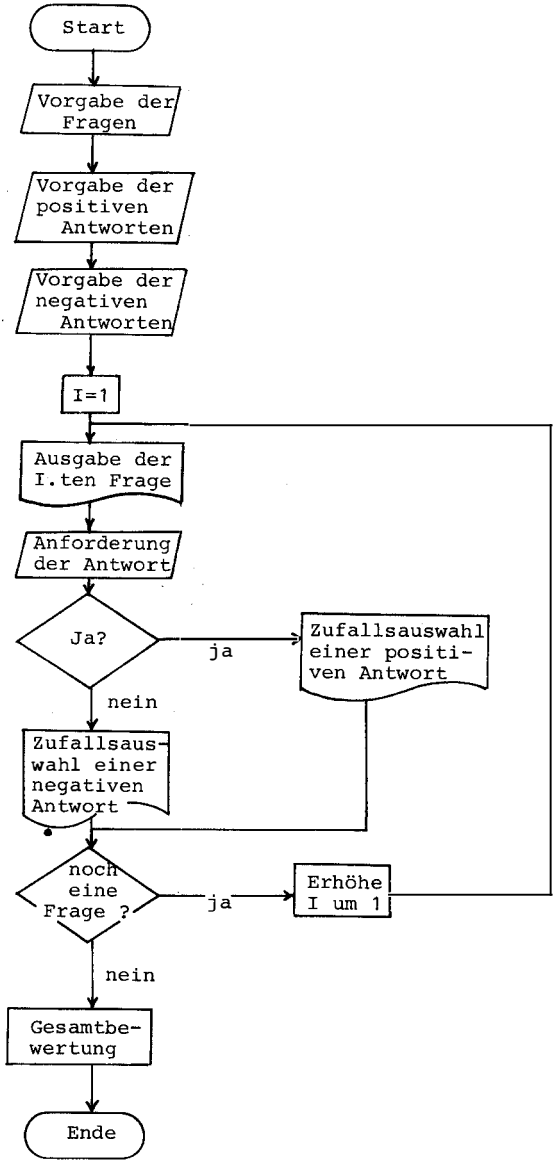
15.4 Problemanalyse

Mit diesen Ausführungen sind die wesentlichen Elemente der problemanalytischen Überlegungen genannt worden :

Zunächst sind die Hintergrundinformationen, die der Rechner für das Gespräch benötigt, einzugeben. Danach wird Frage für Frage bearbeitet, indem das Programm nach dem Zufallsprinzip aus den ihm zur Verfügung stehenden Antworten auswählt; schließlich ist die Schlußbewertung anzufügen.

Wir gelangen hier also zu einem recht einfachen Flußdiagramm zur Beschreibung des geplanten Programmablaufs :

Flußdiagramm : Dialog



15.5 Programm

```
10 REM KI09 - DIALOG
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,    1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 9 :";
60 PRINT "     DIALOG"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM SIMULIERT EIN GESPRAECH"
90 PRINT"MIT DEM RECHNER UEBER DESSEN QUALITAET."
92 PRINT:PRINT:PRINT "BITTE GEBEN SIE IHREN VORNAMEN AN":PRINT
94 PRINTTAB(5)"ICH HEISSE ";:INPUT V$
96 PRINTCHR$(147)
98 PRINTTAB(5)"SEHR GUT, ";V$:PRINT:PRINT:PRINT
100 PRINT"DER RECHNER STELLT IHNEN NUN EINIGE"
105 PRINT"FRAGEN, ";V$:PRINT:PRINT
110 PRINTTAB(5)"BITTE ANTWORTEN SIE NUR MIT"
130 PRINT:PRINTTAB(10)"JA   OOER   NEIN":PRINT:PRINT
140 PRINT"ANDERE ANTWORTEN SIND NICHT ZUGELASSEN."
150 GOSUB 2000:REM WARTEN
160 DIM J$(5),N$(5),F$(10)
170 FOR I=1 TO 5:READ J$(I):NEXT I
190 FOR I=1 TO 5:READ N$(I):NEXT I
190 FOR I=1 TO10:READ F$(I):NEXT I
200 FOR I=1 TO 10
210 PRINTF$(I):PRINT
220 PRINT"JA ODER NEIN (DANN RETURN)";
230 INPUT A$:PRINT:PRINT
240 IF A$<>"JA"AND A$<>"NEIN" THEN PRINT"FALSCHE ANTWORT":PRINT:
    PRINT:GOTO 210
250 IF A$="NEIN" THEN GOTO 300
260 R=INT(RND(1)*5+1)
270 PRINT:PRINTTAB(5)J$(R)
280 PRINT:PRINT:PRINT:K=K+1:GOTO 325
300 R=INT(RND(1)*5+1)
310 PRINT:PRINTTAB(5)N$(R)
320 PRINT:PRINT:PRINT
325 GOSUB 2000:REM WARTEN
330 NEXT I
```

```
340 PRINT:PRINT:PRINT"UEBRIGENS, WIE GEHT ES IHNEN HEUTE,
    ":PRINT
350 PRINT"GUT ODER SCHLECHT ";:INPUT A$
360 IF A$="GUT" THEN PRINT:PRINT"SEHR SCHOEN !":GOTO 392
370 PRINT:PRINT"DAS TUT MIR LEID, DASS ES IHNEN NICHT"
380 PRINT"GUT GEHT."
390 PRINT:PRINT"VIELLEICHT IST ES MORGEN WIEDER BESSER."
392 Z$= " SEHR "
394 IF K<5 THEN Z$=" NICHT SEHR "
396 IF K>4 AND K<8 THEN Z$= " RELATIV "
400 PRINT:PRINT:PRINT:PRINT
410 PRINT"UEBRIGENS, ";V$:PRINT
420 PRINT"SIE HABEN AUF MEINE FRAGEN ";K;" MAL"
430 PRINT"MIT JA GEANTWORTET.":PRINT
440 PRINT"ICH SCHLIESSE DARAUS, DASS SIE MIT MIR"
450 PRINTZ$;" ZUFRIEDEN SIND."
460 PRINT:PRINT:PRINT"ENDE DER AUSGABE":END
1000 REM DATEN
1010 DATA AHA,SEHR SCHOEN,DAS FREUT MICH,PRIMA,GUT
1020 DATA AU WEIA,DAS IST SCHADE,DAS TUT MIR LEID,SOSO,
     WIE SCHLIMM
1030 DATA GEFAELLT IHNEN DIESER RECHNER
1032 DATA IST DIE TASTATUR IN ORDNUNG
1034 DATA REICHT IHNEN DIE SPEICHERKAPAZITAET
1036 DATA BESITZEN SIE EIN FLOPPY-LAUFWERK
1038 DATA HABEN SIE SCHON EINEN DRUCKER
1040 DATA KENNEN SIE DIE DATA-BECKER BUECHER FUER DEN C64
1042 DATA HABEN SIE EINEN FARBMONITOR
1044 DATA IST DAS RECHENTEMPO AUSREICHEND
1046 DATA SIND SIE MIT DER GRAPHIK ZUFRIEDEN
1048 DATA BENUTZEN SIE DEN TONGENERATOR
2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
```

15.6 Variablenliste

A$ = Stringvariable für Tastatureingabe

F$() = Fragen

I = Laufindex

J$() = Positive Antworten

K = Zählindex für positive Antworten

N$() = Negative Antworten

V$ = Vorname des Programmbenutzers

Z$ = Prädikat bei der Endbewertung

15.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tastatureingabe

Satz 80-90 :

Erläuterungen

Satz 92-94 :

Anforderung des Vornamens des Programmbenutzers (siehe auch Satz 98)

Satz 96 :

Löschen des Bildschirms

Satz 98-140 :

Beginn des Gesprächs durch den Rechner mit einer Begrüßung und erklärenden Erläuterungen zum Programmablauf

Satz 150 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tastatureingabe

Satz 160 :

Dimensionierungen für fünf positive und fünf negative Antworten und für zehn Fragen (siehe auch DATA-Statements in Satz 1000-1048)

Satz 170-190 :

Einlesen der Hintergrundinformationen aus den DATA-Statements 1000-1048

Satz 200 :

Beginn der "Fragen-Schleife"

Satz 210-240 :

Ausgabe einer Frage und Anforderung der Antwort des Programmbenutzers (unzulässige Antworten werden in Satz 240 durch Rücksprung zum Satz 210 abgefangen)

Satz 250 :

Wenn der Benutzer "Nein" geantwortet hat, geht es weiter bei Satz 300, andernfalls (Antwort "Ja") bei Satz 260

Satz 260-280 :

Ausgabe einer zufällig ausgewählten positiven Antwort durch den Rechner und Erhöhung des Zählindexes K, der die Anzahl positiver Stellungnahmen des Benutzers zählen soll, um 1;

wenn dies erledigt ist, erfolgt ein Sprung zum Satz 325, d.h. die Reaktionen auf "Nein"-Eingaben (Satz 300-320) werden übersprungen

Satz 300-320 :

Reaktion des Rechners auf "Nein"-Eingaben des Benutzers (ensprechend wie bei "Ja", aber ohne Erhöhung von K)

Satz 325 :

Abwarten; diese Position wird nach einer positiven, wie auch nach einer negativen Reaktion erreicht

Satz 330 :

Übergang zur nächsten Frage (Beendigung der Schleife, die mit Satz 200 beginnt)

Satz 340-350 :

Der Rechner stellt eine Zusatzfrage nach dem Befinden des Benutzers, die mit "gut" oder "schlecht" beantwortet werden kann

Satz 360 :

Antwortet der Benutzer, daß es ihm gut geht, zeigt sich der Rechner erfreut und es erfolgt ein Sprung zum Satz 392

Satz 370-390 :

Geht es dem Benutzer nicht gut, drückt der Rechner sein Bedauern aus und vertröstet auf den morgigen Tag

Satz 392-396 :

Festlegung der Bewertungsprädikate gemäß dem Wert, den die Variable K erzielt hat

Satz 400-450 :

Ausgabe der Bewertung

Satz 460 :

Beendigung des Programms

Satz 1000-1048 :

Ausgangsdatenbestand

Satz 2000-2040 :

Unterprogramm zum Abwarten einer Tastatureingabe (an anderer Stelle schon ausführlich beschrieben)

15.8 Programmergebnisse

Nach dem Starten des Programms fordert dieses nach der Überschrift und Erläuterungen zunächst den Vornamen des Benutzers an.

Geben wir beispielsweise PETER ein, so antwortet das Programm :

SEHR GUT, PETER

Danach folgen weitere Erläuterungen und das Programm stellt seine erste Frage :

GEFAELLT IHNEN DIESER RECHNER

JA ODER NEIN (DANN RETURN)?

Antworten wir beispielsweise mit JA, dann kann (zufallsbedingt) als Antwort erscheinen :

PRIMA

Nach der Fortsetzung des Programms, die wir erreichen, indem wir zum Beispiel einfach die Leertaste drücken, stellt der Rechner die nächste Frage.

Wenn wir diese vielleicht mit NEIN beantworten, dann meldet der Rechner (zufallsabhängig) :

DAS TUT MIR LEID

Wenn alle zehn Fragen gestellt sind, fragt der Rechner :

UEBRIGENS, WIE GEHT ES IHNEN HEUTE,

GUT ODER SCHLECHT ?

Wenn wir mit GUT antworten, meldet der Rechner :

SEHR SCHOEN !

und weiter :

UEBRIGENS, PETER

SIE HABEN AUF MEINE FRAGEN 7 MAL
MIT JA GEANTWORTET.

ICH SCHLIESSE DARAUS, DASS SIE MIT MIR
RELATIV ZUFRIEDEN SIND.

ENDE DER AUSGABE

15.9 Ausblick

Es versteht sich, daß anspruchsvollere Gespräche nur dann in Gang kommen, wenn man aufwendiger programmiert.

Am einfachsten wäre zunächst - um bei obigem Beispiel anzuknüpfen - die Zahl der Fragen und vor allem die Zahl der Antwortmöglichkeiten, aus denen das Programm auswählen kann, drastisch zu erhöhen.

Insbesondere muß aber noch einmal auf die oben erwähnte Möglichkeit hingewiesen werden, daß das Programm auf die Eingaben des Benutzers konkret eingeht, indem es Schlüsselworte aus diesen Eingaben für den Aufbau seiner eigenen Antworten verwendet, anstatt auf "vorgestanzte" Antworten zurückzugreifen.

Weitere Verbesserungen sind z.B. in der Weise vorstellbar, daß das Programm Fragen wiederholt, wenn sie vom Benutzer offenbar nicht verstanden wurden und dergl.

Der Leser kann sich sicherlich leicht eine Vielzahl von Möglichkeiten vorstellen, wie ein derartiges Programm anspruchsvoller und komfortabler gestaltet werden kann und wie insbesondere eine Verbindung zu Experten- oder Auskunftssystemen hergestellt werden könnte.

Kapitel 16 : Reaktion

16.1 Aufgabenstellung

Wir haben schon einleitend darauf hingewiesen, daß wir uns in diesem Buch nicht mit der Robotik beschäftigen, obwohl diese einen sehr wichtigen Teilbereich im Themengebiet der künstlichen Intelligenz darstellt.

Hier soll nun aber ein Beispiel angefügt werden, welches sich gewissermaßen auf der "Nahtstelle" zur Robotik befindet. Diese Position erkärt sich, wenn man bedenkt, welches die Hauptaufgaben der Robotik sind :

In erster Linie geht es darum, eine bewegliche Maschine bzw. bewegliche Einzelteile einer solchen Maschine so zu steuern, daß sie bestimmte, vorher definierte Aufgaben erledigen können. Man denke an die programmgesteuerte Bewegung einer künstlichen Hand, an Greifvorgänge, an programmgesteuerte Ortsveränderungen von Fahrzeugen (automatischer Omnibus) oder an die computergesteuerte Flugbahn unbemannter Raketen, der sog. Marschflugkörper.

Immer geht es bei derartigen Aufgaben darum, die jeweilige Bewegung den im Moment herrschenden Bedingungen anzupassen (ragt ein Berg in die Flugbahn, muß der Marschflugkörper sinnvollerweise seinen Kurs verändern) und auf sich ändernde Bedingungen angemessen zu reagieren.

Dieser Problembereich ist allerdings nicht auf die Steuerung von Bewegungsabläufen beschränkt : Angemessene und gegebenenfalls computergesteuerte Reaktionen sind auch in anderem Zusammenhang vorstellbar : Typische Beispiele dafür sind

thermostatgesteuerte Heizungsanlagen oder automatisierte Mischanlagen und dergl.

Derartige Aufgaben können mit dem Commodore C64, wenn man auf weitere periphere Geräte verzichtet, nicht erledigt werden. Hat man hingegen geeignete periphere Geräte, so sind auch schon mit einem Homecomputer solche Aufgaben simulierbar (z.B. : Computergesteuerte Modelleisenbahnen).

Da wir uns hier aber nur auf die Computer - Grundausstattung beziehen wollen, müssen wir ein "Bewegungsspiel" simulieren und können dabei sehr wohl auch für diesen Anwendungsbereich künstlicher Intelligenz die wesentlichen Grundkonzepte erläutern (siehe dazu Abschnitt 3).

Das wichtigste Problem in einer derartigen Aufgabenstellungen besteht nämlich darin, auf sich verändernde Bedingungen die jeweils angemessenen Reaktionen durch ein dafür geignetes Programm erzeugen zu lassen - und dies kann man auch schon mit der Grundausstattung eines kleinen Homecomputer-Systems darstellen.

16.2 Lösungsansätze

Die Lösung derartiger Steuerungsprobleme ist vom Grundsatz her vergleichsweise einfach :

Wie bei den meisten der anderen besprochenen Problemstellungen auch benötigt das entsprechende Programm eine Informationsbasis, die mindestens aus den beiden folgenden Komponenten bestehen muß:

1. Informationen, die es dann dem Programm ermöglichen, die jeweiligen Umweltbedingungen zu erkennen, auf die angemessen reagiert werden soll:

Das Programm, welches einen Marschflugkörper steuert, muß beispielsweise einen Berg sicher als Hindernis erkennen, welches dann umflogen oder überflogen werden muß, der Robotergreifarm und die Greifhand müssen erkennen, ob ein T-Träger oder ein rohes Ei zu greifen ist, der automatische Omnibus muß feststellen können, on an der nächsten Haltestelle Fahrgäste zu- oder aussteigen wollen usw.

2. Eine Zusammenstellung alternativer Regeln zur Reaktion :

Der Marschflugkörper muß bei einem in der Flugbahn liegenden Berg anders reagieren als bei einer Nebelbank und entsprechendes gilt auch für die anderen oben erwähnten Beispiele.

3. Programmanweisungen zur Durchführung der jeweils angemessenen Reaktion (Steuerprogramm).

4. Programmteile zur Kontrolle :

Es muß jeweils festgestellt werden, inwieweit die jeweilige Reaktion des Programmsystems den Erfordernissen entspricht (Soll-Ist-Vergleich) und

ob Verstärkungen der Reaktion oder eventuell Gegenreaktionen erforderlich sind (man erinnere sich an das Beipiel der automatisch gesteuerten Heizungsanlage per Thermostat).

Wie diese Informationen und Programmteile zusammenwirken, kann an dem folgenden einfachen Beispiel illustriert werden.

16.3 Das Beispiel

Auf dem Bildschirm soll ein sich bewegender Punkt erscheinen (vielleicht erinnert sich der Leser an die ersten, Ende der siebziger Jahre aufkommenden Videospiele, bei denen einfach ein heller Punkt auf dem Bildschirm hin- und herpendelte, der mit einem steuerbaren "Tennisschläger" - ein heller senkrechter Strich auf dem Bildschirm - getroffen werden mußte), der in seiner Bewegung von einem am rechten Bildschirmrand sich auf- und abwärts bewegenden "Fänger-Punkt" aufgefangen werden soll.

Dieses Beispiel illustriert die Programmierung von Reaktionen inswoweit, als der "Fänger" sich auf die Flugbahn des "Balls" einstellen muß, wenn das Auffangen Erfolg haben soll.

Da mit jedem Programmstart der "Ball" von einer Zufallsposition aus gestartet wird, ist in jedem Programmlauf eine andere Reaktion erforderlich.

Hinzu kommt, daß hier auch die Flugrichtung zufallsabhängig ist : Zwar bewegt sich der "Ball" immer von links nach rechts, aber es wird vom Zufall abhängig gemacht, ob er von der zufälligen Startposition sich zunächst nach rechts oben oder nach rechts unten bewegt.

Der "Fänger" muß darüberhinaus auch in Rechnung

stellen, daß der "Ball", wenn dieser den oberen oder den unteren Bildschirmrand erreicht, quasi abprallt, also seine Flugbahn ändert.

Wir werden sehen, daß trotz aller dieser "Erschwernisse", der "Fänger" die Flugbahn des "Balles" korrekt vorherbestimmt und diesen dann zuverlässig "fängt".

16.4 Problemanalyse

Die Problemanalyse ist bei diesem Beispiel wieder einfach :

Zunächst ist per Zufallsauswahl ein Punkt auf dem Bildschirm (also eine Bildschirmspeicheradresse) zu bestimmen, an dem der "Ball" starten soll. Dazu benutzen wir, wie auch schon in anderen Beispielen zuvor, die RND-Funktion.

Den "Ball" selbst zeichnen wir an die jeweilige Bildschirmposition A unter Verwendung folgender Anweisung :

```
POKE A,81
```

Die Codezahl 81 repräsentiert ein helles Kügelchen, unseren "Ball" also.

Desweiteren wird per Zufallsauswahl bestimmt, ob der "Ball" nach rechts oben oder nach rechts unten fliegen soll. Im ersten Fall ist die Zeile der jeweiligen Bildschirmspeicheradresse um den Wert 1 zu verringern, im zweiten Fall ist sie um den Wert 1 zu erhöhen.

Wenn bei dieser Zeilenbestimmung ein Zeilenwert erreicht wird, der über oder unter dem vom Programm auf dem Bildschirm gezeichneten "Rand des Spielfelds" liegt, so ist die Bewegungsrichtung des "Balls" umzukehren, d.h. der Ball prallt gewissermaßen ab.

Es muß nun allerdings beachtet werden, daß wir durch die fortwährende Veränderung der Zeichenadresse A eine Kette von Kügelchen auf dem Bildchirm erzeugen würden, wenn wir nicht dafür sorgten, daß dann, wenn das nächste Kügelchen gezeichnet wird, das vorhergehende gelöscht wird.

Dieses Löschen geschieht mit der Anweisung :

```
POKE AA,32
```

Dabei steht AA für die "alte" Adresse des Bildschirmspeichers (die vorhergehende also) und die Codezahl 32 repräsentiert ein leeres Quadrätchen. Da dieses die gleiche Farbe aufweist wie der Bildschirmhintergrund (und eben leer ist), wird faktisch das Kügelchen an der Position AA gelöscht. Auf diese Weise entsteht im Auge des Betrachters der Effekt eines sich bewegenden Kügelchens, eines "fliegenden Balls".

In dem Moment, in dem der "Ball" startet, startet auch der "Fänger" am rechten Bildschirmrand. Wir verwenden dazu die Anweisung :

```
POKE F,224
```

Dabei steht F für die Bildschirmspeicheradresse des "Fängers" (F bewegt sich in Spalte 36 des Bildschirms in Schrittweiten von 40 Adressen, d.h. verharrt in Spalte 36 und bewegt sich Zeile für Zeile nach oben. Die Codezahl 224 repräsentiert ein helles Quadrätchen (ein inverses leeres Quadrätchen), das unseren "Fänger" darstellen soll.

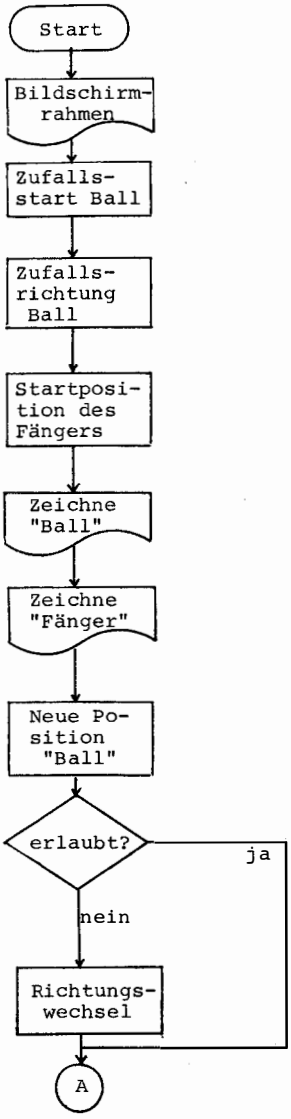
Auch bei der Bewegung des "Fängers" wird an der jeweils vorhergehenden Position gelöscht, um den Bewegungseindruck zu erzeugen.

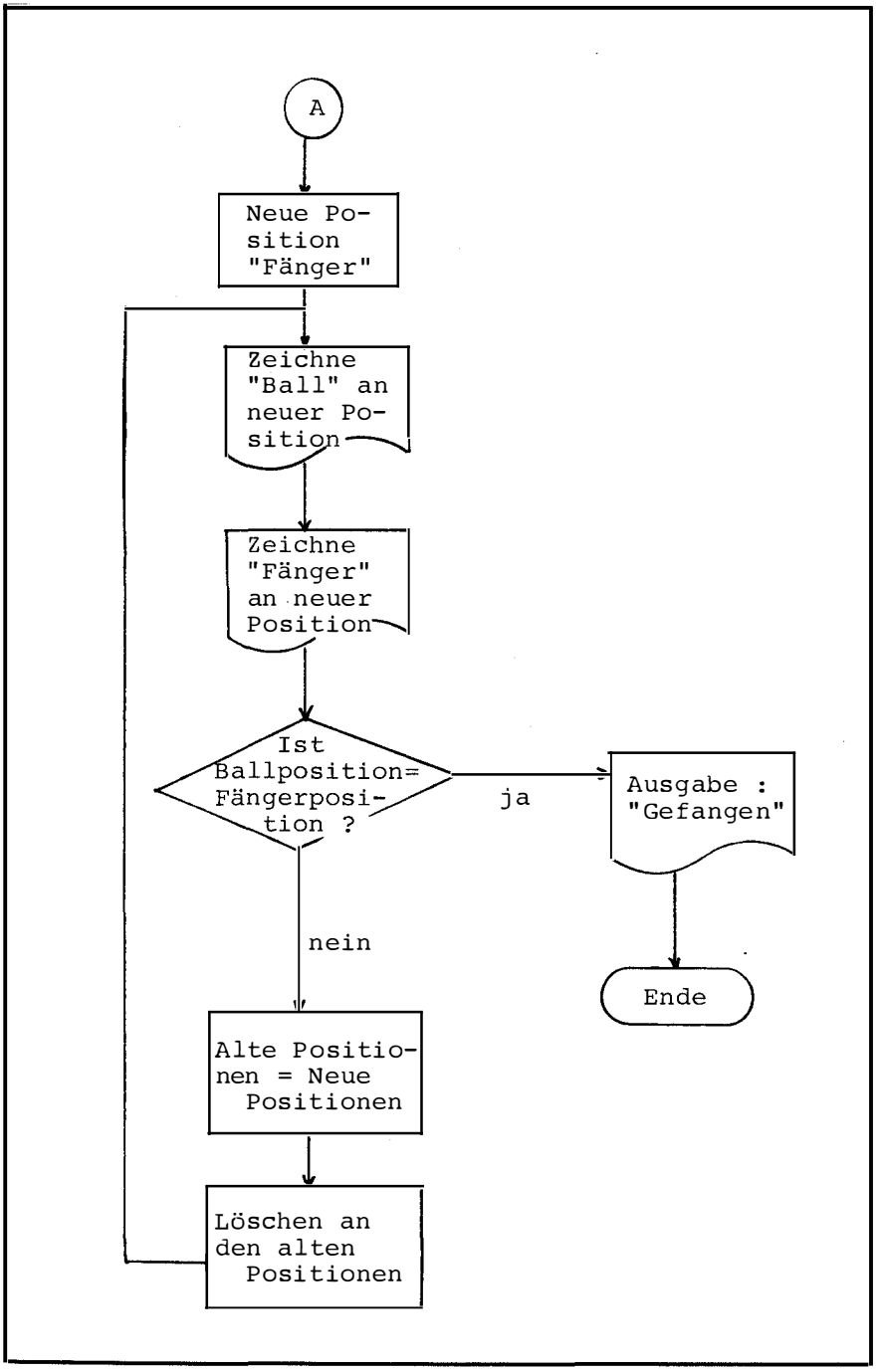
Wenn die Position A des "Balles" mit der Position F des "Fängers" übereinstimmt, so ist der "Ball" erfolgreich durch den "Fänger" aufgefangen worden,

die angemessene Reaktion ist also erfolgt, und das Programm kann damit beendet werden.

Dieser Programmablauf stellt sich schematisch in Form eines Flußdiagramms folgendermaßen dar :

Flußdiagramm "Reaktion"




A
Neue Po-
sition
"Fänger"
Zeichne
"Ball" an
neuer Po-
sition
Zeichne
"Fänger"
an neuer
Position
Ist
Ballposition=
Fängerposi-
tion ?
ja
Ausgabe :
"Gefangen"
nein
Ende
Alte Positio-
nen = Neue
Positionen
Löschen an
den alten
Positionen

16.5 Programm

```
10 REM KI10 - BALL
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,    1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 10 :";
60 PRINT "       BALL"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM SIMULIERT EIN REAGIE-"
90 PRINT"RENDES SYSTEM :":PRINT
100 PRINT"EIN SICH BEWEGENDER PUNKT, DESSEN FLUG-"
110 PRINT"BAHN DURCH EINEN ZUFALLSSTART BESTIMMT"
120 PRINT"WIRD, KANN ZUVERLAESSIG GEFANGEN WERDEN."
130 GOSUB 2000:REM WARTEN
140 REM RAHMEN
150 FOR A=1065 TO 1102:POKE A,64:NEXT A
160 FOR A=1105 TO 1825 STEP 40:POKE A,66:NEXT A
170 FOR A=1142 TO 1822 STEP 40:POKE A,66:NEXT A
180 FOR A=1825 TO 1862:POKE A,64:NEXT A
190 POKE 1065,112:POKE 1102,110
200 POKE 1825,109:POKE 1862,125
210 REM ZUFALLSSTART
220 Z=INT(RND(1)*18+2):S=INT(RND(1)*18+2)
230 A=1024+40*Z+S
240 IF A<1024 OR A>2023 THEN 460
250 R=INT(RND(1)*2)
260 V=1:IF R=0 THEN V=-1
290 F=1024+19*40+36
300 POKE A,81:POKE F,224
310 AA=A:FA=F
320 S=S+ABS(V):Z=Z+V
340 IF Z<3 THEN Z=3:V=-V
350 IF Z>19 THEN Z=19:V=-V
360 A=1024+Z*40+S:F=1024+Z*40+36
370 IF A<1024 OR A>2023 THEN 460
380 IF F<1024 OR F>2023 THEN 460
390 POKE A,81:POKE AA,32
400 POKE F,224:POKE FA,32
410 IF A=F THEN 430
420 GOTO 310
430 PRINT:PRINT:PRINT:PRINT
440 PRINTTAB(15)"GEFANGEN !"
450 END
460 PRINT:PRINT:PRINT"FALSCHE ADRESSE":END
2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
```

16.6 Variablenliste

A$ = Stringvariable zur Tastatureingabe

A = Bildschirmspeicheradresse ("Ball")

AA = Vorhergehende Adresse ("Ball")

F = Bildschirmspeicheradresse ("Fänger")

FA = Vorhergehende Adresse ("Fänger")

R = Zufallszahl

S = Bildschirmspalte

V = Schrittweite bei der Bewegung ("Ball")

Z = Bildschirmzeile

16.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tastatureingabe

Satz 80-120 :

Erläuterungen

Satz 130 :

wie Satz 70

Satz 140-180 :

Zeichnen eines "Spielfeldrandes" (Codezahl 64 repräsentiert einen waagrechten Strich, Codezahl 66 einen senkrechten Strich)

Satz 190-200 :

Zeichnen der Ecken des "Spielfeldes"

Satz 200-230 :

Bestimmung der zufälligen Startposition A des "Balles" (Zeilenposition zwischen 1 und 19, Spaltenposition zwischen 1 und 19); wie man bei gegebener Zeile und Spalte die jeweilige Bildschirmadresse A bestimmt, wurde schon beschrieben (siehe Kapitel "Kunst"))

Satz 250-260 :

Bestimmung der Zufallsrichtung : Entweder gilt V=1 (d.h. Erhöhung des jeweiligen Zeilenwertes) oder V=-1 (d.h. Verminderung)

Satz 290 :

Bestimmung der Startposition des "Fängers"

Satz 300 :

Zeichnen von "Ball" und "Fänger"

Satz 310 :

Deklarierung der jeweiligen Positionen A und F als zukünftig "alte" Positionen AA und FA

Satz 320 :

Bestimmung von Zeile und Spalte der neuen Bildschirmposition des "Balles" bzw. nur der Zeilenposition des "Fängers" (er bleibt ja immer in Spalte 36)

Satz 340-350 :

Überprüfung, ob der "Ball" am "Spielfeldrand" "abprallen" muß; dies ist dann der Fall, wenn die neue Zeile (siehe 320) außerhalb des "Spielfeldes" (siehe Satz 140-200) liegt; es wird dann die Bewegungsrichtung gewechselt

Satz 360 :

Bestimmung der neuen Position des "Balls" A und des "Fängers" F

Satz 370-380 :

Überprüfung, ob eine der beiden Positionen unzulässig, d.h. außerhalb des zulässigen Bildschirmspeicherbereichs liegt; Wenn ja, darf nicht gezeichnet werden, weshalb zum Satz 460 gesprungen wird

Satz 390 :

Zeichnen des "Balls" an die neue Position und Löschen an der alten Position

Satz 400 :

Entsprechend für den "Fänger"

Satz 410 :

Überprüfung, ob die "Ball" - Position A und die "Fänger" - Position F übereinstimmen; wenn ja, erfolgt ein Sprung zum Satz 430

Satz 420 :

Dieser Satz wird erreicht, wenn noch keine Übereinstimmung zwischen A und F erzielt wurde; in diesem Fall muß der "Ball" weiterfliegen, was wir durch einen Rücksprung zum Satz 310 erreichen

Satz 430-450 :

Wenn der "Ball" gefangen wurde (A = F; siehe Satz 410), wird eine Erfolgsmeldung ausgegeben (440) und das Programm beendet (450)

Satz 460 :

Dieser Satz wird nur erreicht, wenn einmal eine Bildschirmspeicheradresse nicht zulässig ist (also nicht zwischen 1024 und 2023 liegt); in diesem Fall wird das Programm mit einer Fehlermeldung beendet (es handelt sich hier gewissermaßen um eine "Notbremse")

Satz 2000-2040 :

Unterprogramm zum Abwarten einer Tastatureingabe

16.8 Programmergebnisse

Wie auch schon bei dem graphisch orientierten Programm "Kunst" ist hier eine Ergebnisdarstellung überflüssig. Wenn der Benutzer das Programm startet, sieht es sehr viel besser, als es hier darstellbar wäre, was geschieht.

16.9 Ausblick

Man kann sich sicherlich leicht vorstellen, wie aus einem so einfachen Bewegungsspiel deutlich kompliziertere Bewegungen erzeugt werden können :

Man könnte zum Beispiel die Bewegungsrichtung und die Bewegungsgeschwindigkeit mit der RND-Funktion beeinflussen, man könnte dafür sorgen, daß der "Fänger" über die Tastatur (oder über einen angeschlossenen Joy-Stick) gesteuert wird, und nicht "quasi-automatisch" den richtigen Weg findet; man könnte mehrere "Bälle" fliegen lassen und auf eventuelle Kollisionen überprüfen; man könnte komplizierte Figuren in ein derartiges Spiel etwa über die SPRITE-Programmierung (siehe C64 - Benutzerhandbuch) einbringen, eventuelle Kollisionen mit einem Explosionsgeräusch koppeln usw.

Auf diese Weise kann man sich Schritt für Schritt an die Programmierung anspruchsvollerer Bewegungsspiele herantasten bis hin zu den sog. Action-Spielen , den Weltraumschlachten u.ä.

Dies alles soll hier jetzt nicht im Detail interessieren. Wichtig war, zu zeigen, wie sich ein sehr einfaches Reaktionsverhalten programmieren läßt, so daß der Eindruck des intelligenten Verhaltens des Programms entstehen kann.

Kapitel 17 : Unterricht

17.1 Aufgabenstellung

In diesem letzten Kapitel wollen wir schließlich zeigen, wie intelligente Programmsysteme dazu verwendet werden können, das Erlernen bestimmter Sachverhalte oder Tatbestände zu erleichtern und somit in der Lage sind, gewissermaßen einen Lehrer zu ersetzen.

Ein solches Programmsystem muß, wie an dieser Einleitung schon erkennbar ist, Elemente von Dialogprogrammen, von Auskunfts- und auch von Expertensystemen und von Reaktionsprogrammen in sich vereinen, so daß dieses letzte Beispiel eigentlich als Zusammenfassung und Wiederholung der vorangegangenen (von einigen Ausnahmen abgesehen) verstanden werden kann.

Es geht bei derartigen Unterrichstprogrammen im einzelnen um die folgenden Aufgabenstellungen :

1. Dem Benutzer werden zunächst Informationen bereitgestellt;

2. Von Fall zu Fall dann wird durch geeignete Fragestellungen überprüft, ob der Benutzer die bislang vorgetragenen Informationen verstanden bzw. im Gedächtnis behalten hat.

3. Ist dies der Fall, kann im Unterrichtsstoff weitergefahren werden; macht der Benutzer hingegen bei der Beantwortung dieser Fragen Fehler, so sind erläuternde Informationen nachzuliefern, bzw. es wird einfach noch einmal der gerade in Frage stehende Stoff wiederholt.

17.2 Lösungsansätze

In diesem Abschnitt sind nun keine zusätzlichen Ausführungen erforderlich. Es wurde ja schon darauf hingewiesen, daß in diesem Beispiel nichts prinzipiell Neues auftritt, so daß wir uns auf die entsprechenden Überlegungen in den vorangegangenen Kapiteln beziehen können.

17.3 Das Beispiel

Als Beispiel für ein Unterrichtsprogramm wählen wir einen Ausschnitt aus einem Programm, mit dessen Hilfe der Benutzer die Programmiersprache BASIC erlernen kann.

Mit Absicht beschränken wir uns dabei auf einen Ausschnitt, weil derartige Programme sehr umfangreich werden, wobei sich aber fortwährend die gleichen Elemente von Programmstrukturen wiederholen. Es genügt deshalb sicherlich, einen Ausschnitt vorzustellen, der dann ohne große Schwierigkeiten vom Leser, wenn er dies für wünschenswert hält, zu einem echten Lernprogramm erweitert werden kann.

Der Ausschnitt, dem wir uns zuwenden wollen, hat mit der Erklärung und Einübung des in der Programmiersprache BASIC so wichtigen PRINT - Statements zu tun.

17.4 Problemanalyse

Aus den vorangegangenen Ausführungen geht deutlich hervor, daß die Problemanalyse bei einem derartigen Programm keine Schwierigkeiten bereiten wird. Wir müssen ja lediglich dafür sorgen, daß dem Progammbenutzer bestimmte Informationen über den Bildschirm angeoten werden, und daß er diese in Ruhe studieren kann.

Wenn ein bestimmter Informationsvorrat geschaffen ist, kann durch geeignete Abfragen überprüft werden, ob der Benutzer, das, was er gelesen hat, auch behalten hat.

Wenn er auf die vom Programm gestellten Fragen korrekt antwortet, kann im Informationsangebot weiter fortgefahren werden; antwortet er nicht korrekt, so gibt es mehrere Möglichkeiten :

1. Rücksprung im Programm an diejenige Stelle, an der die in Frage stehenden Sachverhalte erläutert wurden, also Wiederholung eines bestimmten Programmsegments;

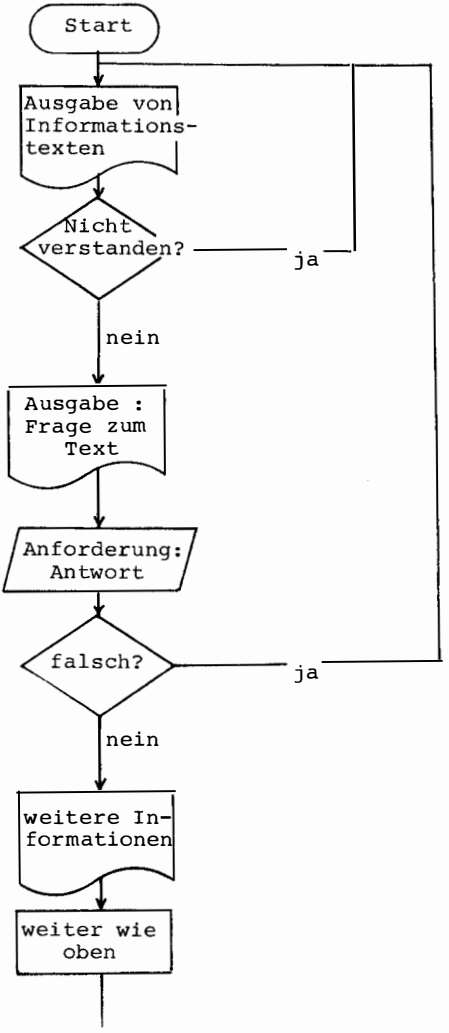
2. Aufforderung, nochmals die Frage und die Antwort genau zu überlegen, um gegebenenfalls einen weiteren Antwortversuch zu wagen;

3. Ausgabe zusätzlicher Erläuterungen und eventuell erneute Ausgabe der zunächst falsch beantworteten Frage.

Man erkennt schon an diesen wenigen Ausführungen, daß ein derartiges Programm in erster Linie aus PRINT-Anweisungen besteht und deshalb auch recht umfangreich wird. Verkürzungen, auf die wir aber im folgenden Programm weitgehend verzichtet haben, sind möglich, wenn man bestimmte, sehr häufig wiederkehrende Worte oder Redewendungen in Stringvariablen speichert und nur diese dann jeweils in den PRINT-Anweisungen benutzt.

Ansonsten stellen sich in einem derartigen Programm keine weiteren Probleme, so daß wir zu dem folgenden Programmablaufplan gelangen :

Flußdiagramm "Unterricht"



17.5 Programm

```
10 REM KI11 - UNTERRICHT
20 PRINTCHR$(147)
30 PRINTTAB(8)"KUENSTLICHE INTELLIGENZ":PRINT:PRINT
40 PRINTTAB(8)"PROF.DR.W.VOSS,     1985":PRINT:PRINT
50 PRINT:PRINT:PRINTTAB(8)"BEISPIEL 11 :";
60 PRINT "UNTERRICHT"
70 GOSUB 2000:REM WARTEN
80 PRINT"DIESES PROGRAMM IST EIN AUSSCHNITT AUS"
90 PRINT"EINEM UNTERRICHTSPROGRAMM ZUR UEBUNG"
100 PRINT"DER PROGRAMMIERSPRACHE   B A S I C."
110 GOSUB 2000:REM WARTEN
120 DIM K$(10)
130 FOR I=1 TO 10:READ K$(I):NEXT I
140 PRINTCHR$(147):PRINTTAB(5)"UEBERSICHT":PRINT
150 FOR I=1 TO 10:PRINTI;TAB(5)K$(I):NEXT I
160 PRINT:PRINT"ACHTUNG :"
170 PRINT"IN DIESEM PROGRAMM IST NUR KAP.3 PRO-"
180 PRINT"GRAMMIERT !"
190 PRINT:INPUT "WELCHES KAPITEL (BITTE ZAHL)";K
200 IF K>10 THEN PRINT"EINGABEFEHLER":PRINT:PRINT:GOTO 150
210 ON K GOSUB 3000,3500,4000,4500,5000,5500,6000,6500,7000,7500
220 PRINT:PRINT"NOCH EIN KAPITEL ";:INPUT "(J/N)";A$
230 IF A$="J" THEN 140
240 PRINT:PRINT"ENDE DES PROGRAMMS":END
2000 REM UP WARTEN
2010 PRINT:PRINT:PRINT
2020 PRINT:PRINT:PRINTTAB(6)"BITTE EINE TASTE DRUECKEN !"
2030 GET A$:IF A$="" THEN 2030
2040 PRINTCHR$(147):RETURN
```

```
2500 REM DATEN
2510 DATA GRUNDBEGRIFFE,SPRACHEN
2515 DATA ERSTE PROGRAMME (PRINT/END/RUN)
2520 DATA DATENEINGABE (LET/INPUT) UND LIST
2530 DATA PROGRAMMSPRUENGE (GOTO/IF...THEN)
2540 DATA SCHLEIFEN (FOR...TO/NEXT)
2550 DATA LESEN (DATA UND READ)
2560 DATA DIMENSIONIERUNGEN (DIM)
2570 DATA UNTERPROGRAMME (GOSUB/RETURN)
2580 DATA STRINGBEARBEITUNG
3000 REM UP KAP.1 ********************
3010 PRINTCHR$(147)
3020 PRINT"KAPITEL 1 IST NOCH NICHT PROGRAMMIERT"
3030 GOSUB 2000
3040 RETURN
3500 REM UP KAP.2 ********************
3510 PRINTCHR$(147)
3520 PRINT"KAPITEL 2 IST NOCH NICHT PROGRAMMIERT"
3530 GOSUB 2000
3540 RETURN
```

```
4000 REM UP KAP.3 ********************
4010 PRINTCHR$(147)
4020 PRINT"DER WICHTIGSTE WEG DER INFORMATIONSAUS-"
4022 PRINT"GABE DURCH EIN LAUFENDES BASIC-PROGRAMM"
4026 PRINT"IST DIE BENUTZUNG DER PRINT-ANWEISUNG"
4028 PRINT:PRINT
4030 PRINT"SIE LAUTET IN ALLGEMEINER FORM :":PRINT
4032 PRINT:PRINT:PRINT
4034 PRINT"   NN PRINT WERT":PRINT:PRINT
4036 GOSUB 2000
4038 PRINT"   NN PRINT WERT":PRINT:PRINT
4040 PRINT"AN DER STELLE   WERT   KANN STEHEN :":PRINT
4042 PRINT"   -  EINE ZAHL"
4044 PRINT"   -  EINE RECHNUNG"
4046 PRINT"   -  EIN STRING (ZEICHENKETTE)"
4048 PRINT"   -  EIN VARIABLENNAME"
4049 PRINT
4050 PRINT"   PRINT KANN AUCH ALLEIN STEHEN"
4051 GOSUB 2000
4052 PRINT:PRINT"BEISPIELE :":PRINT
4054 PRINT"10 PRINT 3":PRINT
4056 PRINT"DIE ZAHL 3 WIRD AUSGEGEBEN"
4068 PRINT:PRINT"20 PRINT 3+5":PRINT
4070 PRINT"DAS RECHNERGEBNIS 8 WIRD AUSGEGEBEN"
4071 GOSUB 2000
4072 PRINT:PRINT"30 PRINT":PRINT
4074 PRINT"EINE LEERE ZEILE WIRD AUSGEGEBEN"
4076 PRINT:PRINT"40 PRINT X":PRINT
4078 PRINT"DER INHALT DES SPEICHERFELDES X WIRD"
4080 PRINT"AUSGEGEBEN"
4082 PRINT:PRINT"SOLL MIT DER PRINT-ANWEISUNG EIN STRING"
4083 PRINT"AUSGEGEBEN WERDEN, SO IST DIESER IN"
4084 PRINT"ANFUEHRUNGSZEICHEN EINZUSCHLIESSEN !"
4086 GOSUB 2000
```

```
4088 PRINT"AUFGABE 1":PRINT:PRINT
4090 PRINT"WIEVIELE MOEGLICHKEITEN GIBT ES, DIE"
4092 PRINT"STELLE   WERT   HINTER DEM ANWEISUNGS-"
4094 PRINT"WORT PRINT ZU FUELLEN ?":PRINT
4096 INPUT "ANZAHL DER MOEGLICHKEITEN : ";Z
4098 IF Z=5 THEN PRINT:PRINT:PRINTTAB(10)"SEHR GUT !":GOTO 4108
4100 PRINT:PRINT:PRINT"DAS IST NICHT RICHTIG.":PRINT
4102 PRINT"BETRACHTEN SIE NOCH EINMAL DIE VER-"
4104 PRINT"SCHIEDENEN MOEGLICHKEITEN UND DIE BEI-"
4106 PRINT"SPIELE DAZU : ":GOSUB 2000 :GOTO 4038
4108 PRINT:PRINT:PRINT
4110 PRINT"BITTE BEANTWORTEN SIE DIE FOLGENDEN "
4112 PRINT"FRAGEN NUR MIT J (FUER JA) ODER N (FUER"
4114 PRINT"NEIN)":PRINT:PRINT
4116 B$="IST DIESE AUSSAGE RICHTIG (J/N)"
4118 PRINT"DIE ANWEISUNG   ";
4120 PRINT"100 PRINT 3+12/3":PRINT
4122 PRINT"ERGIBT DEN WERT 7":PRINT
4124 PRINTB$;:INPUT A$
4126 IF A$="J" THEN PRINT:PRINTTAB(7)"SEHR GUT !":GOTO 4140
4128 PRINT:PRINT"DAS IST NICHT KORREKT"
4130 PRINT:INPUT "NOCH EIN VERSUCH (J/N)";A$
4132 IF A$="J" THEN 4118
4134 PRINT:PRINT"DAS RICHTIGE ERGEBNIS IST IN DER TAT  7"
4136 PRINT:PRINT"BITTE NOCH EINMAL DIE GRUNDLAGEN DER"
4138 PRINT"SCHULMATHEMATIK NACHARBEITEN !":PRINT
4140 PRINT:PRINT
4142 PRINT"SOLL MIT DER PRINT-ANWEISUNG EIN STRING"
4144 PRINT"AUSGEGEBEN WERDEN, SO IST DIESER IN"
4146 PRINT"ANFUEHRUNGSZEICHEN EINZUSCHLIESSEN":PRINT
4148 PRINTB$;:INPUT A$
4150 IF A$="J" THEN PRINT:PRINTTAB(7)"SEHR GUT !":GOTO 4160
4152 PRINT:PRINT"IHRE ANTWORT IST NICHT RICHTIG":PRINT
4154 PRINT"DESHALB WOLLEN WIR NOCH EINMAL ALLES"
4156 PRINT"WIEDERHOLEN, WAS BISLANG ZUR PRINT-AN-"
4158 PRINT"WEISUNG GESAGT WURDE.":PRINT:PRINT:GOTO4020
4160 PRINT:PRINT
4162 PRINT"DIE ANWEISUNG     ";
4164 PRINT"120 PRINT A":PRINT
4166 PRINT"ERGIBT AUF DEM BILDSCHIRM DAS SYMBOL  A":PRINT
4168 PRINTB$;:INPUT A$
4170 IF A$="N" THEN PRINT:PRINTTAB(7)"SEHR GUT !":GOTO 4182
4172 PRINT:PRINT"IHRE ANTWORT WAR NICHT KORREKT":PRINT
4174 PRINT"DIE OBIGE ANWEISUNG ERGIBT DEN INHALT"
4176 PRINT"DES SPEICHERFELDES A AUF DEM BILDSCHIRM"
4178 PRINT:PRINT"WENN A NOCH NICHT BESETZT WAR, ERGIBT"
4180 PRINT"SICH EINE  0 ."
4182 GOSUB 2000
```

```
4184 PRINT"NUN EINIGE WEITERE INFORMATIONEN :":PRINT
4185 PRINT"DIE MOEGLICHKEITEN, DIE AN DER STELLE"
4186 PRINT"WERT   IN DER PRINT-ANWEISUNG STEHEN"
4188 PRINT"KOENNEN, SIND AUCH MEHRFACH UND/ODER"
4190 PRINT"GEMISCHT VERWENDBAR.":PRINT
4192 PRINT"SIE MUESSEN DANN ABER DURCH STRICHPUNKT"
4194 PRINT"ODER DURCH KOMMA VONEINANDER GETRENNT"
4196 PRINT"WERDEN.":GOSUB 2000
4198 PRINT"BEISPIEL :":PRINT
4200 PRINT"130 PRINT 3+5,7-2,8*2"
4202 PRINT:PRINT"DIESE ANWEISUNG ERGIBT :":PRINT
4204 PRINT"  8          5         16"
4206 PRINT:PRINT"BENUTZEN WIR IN DER GLEICHEN ANWEISUNG"
4208 PRINT"DEN STRICHPUNKT ALS TRENNER, SO ERHAL-"
4210 PRINT"TEN WIR : ":PRINT
4212 PRINT"  8  5  16"
4214 GOSUB 2000
4216 PRINT"AN DIESER STELLE HABEN WIR DIE PROGRAM-"
4218 PRINT"MIERUNG ABGEBROCHEN, WEIL DAS PRINZIP"
4220 PRINT"JETZT KLAR SEIN DUERFTE. (4216-4220)"
4222 GOSUB 2000
4480 PRINT:PRINT"ENDE DES KAPITELS 3":GOSUB 2000
4490 RETURN
4500 REM UP KAP.4 ********************
4510 PRINTCHR$(147)
4520 PRINT"KAPITEL 4 IST NOCH NICHT PROGRAMMIERT"
4530 GOSUB 2000
4540 RETURN
```

```
5000 REM UP KAP.5 ********************
5010 PRINTCHR$(147)
5020 PRINT"KAPITEL 5 IST NOCH NICHT PROGRAMMIERT"
5030 GOSUB 2000
5040 RETURN
5500 REM UP KAP.6 ********************
5510 PRINTCHR$(147)
5520 PRINT"KAPITEL 6 IST NOCH NICHT PROGRAMMIERT"
5530 GOSUB 2000
5540 RETURN
6000 REM UP KAP.7 ********************
6010 PRINTCHR$(147)
6020 PRINT"KAPITEL 7 IST NOCH NICHT PROGRAMMIERT"
6030 GOSUB 2000
6040 RETURN
6500 REM UP KAP.8 ********************
6510 PRINTCHR$(147)
6520 PRINT"KAPITEL 8 IST NOCH NICHT PROGRAMMIERT"
6530 GOSUB 2000
6540 RETURN
7000 REM UP KAP.9 ********************
7010 PRINTCHR$(147)
7020 PRINT"KAPITEL 9 IST NOCH NICHT PROGRAMMIERT"
7030 GOSUB 2000
7040 RETURN
7500 REM UP KAP.10 ********************
7510 PRINTCHR$(147)
7520 PRINT"KAPITEL 10 IST NOCH NICHT PROGRAMMIERT"
7530 GOSUB 2000
7540 RETURN
```

17.6 Variablenliste

A$ = Stringvariable zur Tastatureingabe (z.B. J für Ja und N für Nein)

B$ = Hilfsstring bei der Formulierung von Fragen

I = Laufindex

K = Nummer des ausgewählten Kapitels

K$() = Kapitelüberschriften

Z = Speicherfeld für numerische Eingaben

17.7 Programmbeschreibung

Satz 10-60 :

Überschrift

Satz 70 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tastatureingabe

Satz 80-100 :

Erläuterungen

Satz 110 :

wie Satz 70

Satz 120-130 :

Dimensionierung und Einlesen der Überschriften der

Kapitel (siehe auch Satz 2500-2580)

Satz 140-150 :

Ausgabe der Kapitelübersicht

Satz 160-180 :

Erläuterung zur Programmierung

Satz 190 :

Anforderung des Kapitels, für das sich der Benutzer interessiert (in diesem Programm liegt nur ein Ausschnitt des Kapitels 3 vor)

Satz 200 :

Unzulässige Kapitelwünsche werden abgefangen und eine Neueingabe durch Rücksprung zum Satz 150 ermöglicht

Satz 210 :

In Abhängigkeit von der Eingabe des Benutzers wird mit diesem Satz in verschiedene Unterprogramme gesprungen (pro Kapitel ist ein Unterprogramm vorgesehen)

Satz 220-230 :

Nach dem erfolgten Rücksprung aus dem jeweiligen Unterprogramm wird dieser Satz erreicht, der es dem Benutzer ermöglicht, ein weiteres Kapitel (das aber, wie schon erwähnt wurde, nicht programmiert vorliegt) in Angriff zu nehmen;

wenn er dies wünscht, erfolgt ein Rücksprung zum Satz 140, d.h. es wird erneut die Kapitelübersicht ausgegeben, so daß der Benutzer wieder auswählen kann

Satz 240 :

Wünscht der Benutzer keine Wiederholung, so wird das Programm beendet

Satz 2000-2040 :

Unterprogramm zum Abwarten einer Tastatureingabe, wie es an anderer Stelle schon erläutert wurde

Satz 2500-2580 :

Daten (Kapitelüberschriften)

Satz 3000-3540 :

Unterprogramme für Kapitel 1 und Kapitel 2 des Unterrichtsprogramms; diese Kapitel sind aber nicht programmiert; deshalb erfolgt in diesen Programmsegmenten nur eine entsprechende Ausgabe und dann sofort der Rücksprung in das rufende Hauptprogramm

Satz 4000 :

Beginn des Unterprogramms, in dem der Anfang des Kapitels 3 programmiert ist

Satz 4010-4086 :

Ausgabe der ersten Informationstexte; da diese Texte relativ rasch den Bildschirm füllen und dann über den oberen Bildschirmrand hinauswandern, muß an geeigneten Stellen das Programm durch Sprung ins Unterprogramm 2000 (Abwarten) unterbrochen werden (4051, 4071, 4086)

Satz 4088-4096 :

Hier wird nun eine erste Kontrollfrage gestellt und die Antwort des Benutzers in Satz 4096 angefordert

Satz 4098 :

Ist diese Antwort korrekt, wird der Benutzer gelobt und es erfolgt ein Programmsprung zum Satz 4108; ist sie hingegen nicht korrekt, folgt Satz 4100

Satz 4100-4106 :

Dem Benutzer wird mitgeteilt, daß seine Antwort nicht korrekt war und es wird zurückgesprungen zum Satz 4038, um die fraglichen Programmsegmente erneut zu durchlaufen

Satz 4108-4116 :

Vorbereitungen für eine Serie von Fragen, die nun folgen, und die der Benutzer jeweils mit "Ja" oder "Nein" (J oder N) beantworten soll

Satz 4118-4122 :

Erste Frage

Satz 4124-4126 :

Anforderung der Antwort; ist diese korrekt, erfolgt ein Sprung zum Satz 4140

Satz 4128-4130 :

Dem Benutzer wird mitgeteilt, daß seine Antwort nicht korrekt war und das Programm fragt ihn, ob er einen weiteren Antwortversuch wagen will; wenn ja, springt das Programm zurück zu Satz 4118, d.h. die Frage wird erneut gestellt

Satz 4134-4138 :

Will der Benutzer keinen weiteren Antwortversuch wagen, wird das korrekte Ergebnis ausgegeben

Satz 4140-4148 :

Formulierung der zweiten Frage und Anforderung der Antwort
Satz 4150 :

Bei korrekter Antwort wird zum Satz 4160 verzweigt

Satz 4152-4158 :

Ist die Antwort hingegen nicht korrekt, wird dies dem Benutzer mitgeteilt und es erfolgt der Rücksprung zum Satz 4020, d.h. der gesamte bisherige Stoff wird wiederholt (hier bieten wir also abwechslungshalber dem Benutzer nicht die Möglichkeit, einen weiteren Antwortversuch zu wagen; aber selbstverständlich wäre auch eine derartige Programmierung wie oben möglich)

Satz 4160-4168 :

Formulierung der dritten Frage und Anforderung der Antwort

Satz 4170 :

Ist die Antwort korrekt, erfolgt eine Verzweigung zum Satz 4182

Satz 4172-4180 :

Bei nicht korrekter Antwort wird die richtige Antwort nebst einer Erläuterung durch das Programm ausgegeben; also auch hier keine Möglichkeit, eine weitere Antwort zu wagen und hier nun auch keine Stoffwiederholung, weil diese sich bei dieser dritten Frage nicht anbietet

Satz 4182 :

Sprung ins Unterprogramm 2000 zum Abwarten einer Tastatureingabe

Satz 4184-4214 :

Weitere Informationen, wieder an geeigneten Stellen durch Programmunterbrechungen per Unterprogramm 2000 unterbrochen
Satz 4216-4222 :

Abbruchhinweis

Satz 4480-4490 :

Beendigung der Ausgaben für Kapitel 3 und Rücksprung ins rufende Hauptprogramm

Satz 4500-7540 :

Unterprogramme für die Kapitel 4 bis 10, die aber ebenfalls wie jene der Kapitel 1 und 2 nicht ausgeführt sind

17.8 Programmergebnisse

Auf die Darstellung von Programmergebnissen kann hier wieder verzichtet werden. Wir hätten ja nichts anderes darzustellen, als die einzelnen Mitteilungen, Fragen und Antworten des Gesprächs, das sich mit Hilfe des obigen Programms zwischen Benutzer und Rechner ergibt.

Genau dieses Gespräch aber kann man sehr viel besser verfolgen, wenn man das Programm einfach mit dem Kommando RUN startet.

17.9 Ausblick

Auch in diesem letzten Abschnitt ist nicht viel zu sagen. Wir haben schon darauf hingewiesen, daß derartige Programme im Zusammenhang mit Auskunfts- und Expertensystemen eine große Bedeutung erlangen können und es versteht sich, daß insbesondere im schulischen Bereich derartige Unterrichtsprogramme eine wirksame Unterstützung für die Bemühungen der Lehrer sein können.

Ohne Schwierigkeiten wird der Leser erkennen, daß nach dem Muster des obigen Programms entsprechende "Gespräche" für fast alle schulischen Fächer programmiert werden können - und natürlich auch für Wissensgebiete außerhalb des schulischen Bereichs.

Kapitel 18 : Weitere Programme

In den vorangegangenen elf Kapiteln haben wir je ein Demonstrationsbeispiel zu den wichtigsten Anwendungsbereichen der sogenannten künstlichen Intelligenz vorgestellt.

Bei diesen Beispielen stand nicht ihre praktischen Verwendungsmöglichkeiten im Mittelpunkt des Interesses, sondern sie wurden vor allem unter dem Gesichtspunkt entwickelt, daß in möglichst überschaubarer und anschaulicher Weise dem Leser die Grundprinzipien solcher Programme und der entsprechenden Lösungswege verdeutlicht werden konnten.

Um weitere Beispiele, vor allem solcher, die eher als anwendungsorientiert bezeichnet werden können, vorstellen zu können, fehlte im Rahmen dieser Veröffentlichung der Platz.

Aus diesem Grund haben wir eine Diskette zu diesem Buch vorbereitet, auf der sich alle die hier vorgestellten Programme befinden und darüberhinaus weitere aus den gleichen Anwendungsbereichen der künstlichen Intelligenz.

Die zusätzlichen Programme auf dieser Diskette sind in der Regel ähnlich strukturiert, wie die hier vorgestellten Programme, so daß der Benutzer dieser Diskette nach der Lektüre dieses Buches sicherlich auch diese Zusatzprogramme verstehen kann, obgleich diese nicht mehr im Detail beschrieben sind.

Es handelt sich bei diesen Zusatzprogrammen um die folgenden Aufgabenstellungen :

1. Expertensysteme :

1.1 Einfaches medizinisches Diagnosesystem

1.2 Diätberatungen

1.3 Beratungen für Immobilien-Interessierte

2. Auskunftssysteme :

2.1 Schallplattenarchiv

2.2 Sammlung von Kochrezepten

2.3 Mathematische Tafeln

3. Suchprogramme

3.1 Primzahlen-Suchprogramm

3.2 Aufsuchen bestimmter Zahlenwerte und Sortieralgorithmus

3.3 Auswertung von Texten

4. Entscheiden

4.1 Prüfung von Hypothesen über Mittelwerte

4.2 Unabhängigkeitstest

4.3 Hochrechnungen

5. Erkennen

5.1 Übersetzungen

5.2 Erkennen von Buchstaben

5.3 Erkennen von geometrischen Figuren

6. Spielprogramme

6.1 "17 und 4"

6.2 Intelligente Lotto-Tips

6.3 Groschenautomat

7. Selbstlernende Systeme

7.1 Labyrinth

7.2 Identifizieren geometrischer Figuren

7.3 Verbesserungen medizinischer Diagnosen

8. Kunst

8.1 Bewegte Graphik

8.2 Computergedichte

8.3 Zufallsmuster

9. Dialoge

9.1 Antwortprogramm

9.2 Gesprächstherapie

9.3 Fahrplanauskünfte

10. Reaktionen

10.1 Kamera-Automatik

10.2 Heizungs-Thermostat

10.3 Feedback-System

11. Unterrichtsprogramme

11.1 Großes Einmaleins

11.2 Vokabeltest

11.3 Historische Daten

Der Leser erkennt an diesem unkommentierten Überblick, daß auch diese zusätzlichen Programme noch zum großen Teil demonstrativen Charakter aufweisen.

Auf diese Weise wird es möglich, daß der Programmbenutzer diese Programme noch relativ leicht nachvollziehen kann, um somit in die Lage versetzt zu werden, anspruchsvollere Programme nach seinen eigenen Bedürfnissen zu entwickeln, wenn er dies wünscht.

Literaturhinweise

Die folgende Literaturübersicht erhebt keinen Anspruch auf Vollständigkeit. Vielmehr soll mit dieser Übersicht erreicht werden, daß der Leser die hier angesprochenen Themen vertieft studieren kann, daß ihm weiterführende Literatur geboten wird, daß er Ergänzungen und Erweiterungen finden kann.

Wir haben diese Literaturliste deshalb auch kapitelweise geordnet, was notwendigerweise zu einigen Doppelnennungen führt. Diese Literatur wurde unter anderem auch unter dem Gesichtspunkt ausgewählt, daß dort wiederum weiterführende Literaturhinweise zu finden sind.

Kap. 1 : Grundbegriffe

BÜHLER,C. (1962) : Psychologie im Leben unserer Zeit, München/Zürich

DOBZHANSKY,T. (1975) : Evolutionary Roots of Familiy Ethics and Group Ethics, in : The Centrality of Science and Absolute Values, Band I, Proceedings of the Fourth International Conference on the Unity of Sciences, New York, S. 411 ff.

GOLDSTEIN,I.F. / GOLDSTEIN,M. (1978) : How we know, An Exploration of the Scientific Process, New York

GOLDSTEIN,I. / PAPERT,S. (1977) : Artificial Intelligence, Language and the Study of Knowledge, in : Cognitive Science, Band 1, Nr, 1, Seite 84 ff.

GRAHAM,N. (1983) : Künstliche Intelligenz, wie Sie Ihren Computer zum Denken bringen, Sprendlingen

HARTNELL,T. (1984) : Exploring Artificial Intelligence on your Microcomputer, London / Melbourne

HOFSTÄTTER,P.R. (1957) : Informationstheorie, in : Psychologie (Hg.: P.R.Hofstätter), Frankfurt

HOFSTÄTTER,P.R. (1957) : Intelligenz, in : Psychologie (Hg.: P.R.Hofstätter), Frankfurt

KENT,E.W. (1981) : The brains of men and machines, New York

KOBSA,A. (1984) : Artificial Intelligence und Kognitive Psychologie, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 99 ff.

LORENZ,K. (1982) : Die Rückseite des Spiegels, Versuch einer Naturgeschichte menschlichen Erkennens, 6.Auflage, München

NEUMANN, von, J. (1958) : The Computer and the brain, New Haven / London

POPPER,K.R. / ECCLES,J.C. (1982) : Das Ich und sein Gehirn, 2.Auflage, München

RIEDL,R. (1980) : Biologie der Erkenntnis, die stammesgeschichtlichen Grundlagen der Vernunft, Berlin / Hamburg

RITCHIE,D. (1984) : Gehirn und Computer, Stuttgart

SCHRÖDINGER,E. (1951) : Was ist Leben ?, München

SHANNON,C.E. / WEAVER,W. (1949) : The mathematical theory of communication

SIMONS,G. (1984) : Sind Computer lebendig ? Stand und Zukunft der Computerentwicklung, München

STEINACKER,I. (1984) : Intelligente Maschinen ?, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 7 ff.

TRAPPL,R. (1984) : Auswirkungen der Artificial Intelligence, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 199 ff.

TEILHARD DE CHARDIN,P. (1981) : Der Mensch im Kosmos, München

WEISS,P. (Hg) (1971) : Hierarchically organized systems in theory and practice, New York

WEIZENBAUM,J. (1978) : Die Macht der Computer und die Ohnmacht der Vernunft, Frankfurt

WEIZENBAUM,J. (1984) : Kurs auf den Eisberg, Zürich

Kap. 2 : Ein Blick in die Vergangenheit

EURICH,C. (1980) : Das verkabelte Leben, wem schaden und wem nützen die neuen Medien ?, Reinbek

FRIEDRICHS,G. / SCHAFF,A. (Hg.) (1982) : Auf Gedeih und Verderb, Mikroelektronik und Gesellschaft, Bericht an den Club of Rome, Wien / München / Zürich

MÜLLERT,N. (Hg.) (1982) : Schöne elektronische Welt, Computer - Technik der totalen Kontrolle, Reinbek

SIEMENS AG (1984) : Chancen mit Chips, Zwischenbilanz einer Basistechnologie, Berlin / München

WOLTERS,M.F. (Hg.) (1974) : Der Schlüssel zum Computer, Reinbek

Kap. 3 : Anwendungsbereiche der künstlichen Intelligenz

HORN,W. (1984) : Methoden der Artificial Intelligence, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 27 ff.

RETTI,J. (1984) : Knowledge Engineering und Expertensysteme, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 73 ff.

RITCHIE, D. (1984) : Gehirn und Computer, Stuttgart

STEINACKER,I. (1984) : Intelligente Maschinen ? in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 7 ff.

TRAPPL,R. (1984) : Auswirkungen der Artificial Intelligence, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 199 ff.

Kap. 4 : Zur Funktionsweise moderner Rechner

BARTEL / JORDAN (1984) : Data Becker`s Homecomputer-Buch, Düsseldorf

OSBORNE,A. (1978) : Mikrocomputer-Grundwissen, eine allgemeinverständliche Einführung in die Mikrocomputer-Technik, München

SACHT,H.-J. (1981) : Tischcomputer für Heim und Beruf

VOSS,W. (1984) : EDV und BASIC, Mikrocomputer-Grundwissen, Skriptum, 2.Auflage, Bochum

Kap. 5 : BASIC-Grundlagen

KAMPOW (1984) : Das BASIC Trainingsbuch zum Commodore 64, Düsseldorf

POL,B. (1984) : Wie man in BASIC programmiert, Einführung, Techniken, Fallstudien, Würzburg

VOSS,W. (1984) : Grundelemente der Programmiersprache BASIC, Skriptum, Bochum

VOSS,W. (1984) : Das Schulbuch zum Commodore 64, Düsseldorf

Kap. 6 : Einführung

BRAIN,K. / BRAIN,S. (1984) : Artificial Intelligence on the commodore 64, make your micro think, Exeter

HAMBUSCH,R. (1974) : Organisationslehre, Darmstadt

STEDE,M. u.a. (1984) : Einführung in die künstliche Intelligenz, Band 2 : Anwendungsgebiete, Sprendlingen

Kap.7 : Expertensystem

BRAIN,K. / BRAIN,S. (1984) : Artificial Intelligence on the commodore 64, make your micro think, Exeter (Kap.5)

RETTI,J. (1984) : Knowledge Engineering und Expertensysteme, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 73 ff.

STEDE,M. u.a. (1984) : Einführung in die künstliche Intelligenz, Band 2 : Anwendungsgebiete, Sprendlingen (Kap.1)

STEINACKER,I. (1984) : Intelligente Maschinen ?, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 7 ff.

Kap. 8 : Auskunftssystem

GRAHAM,N. (1983) : Künstliche Intelligenz, wie Sie Ihren Computer zum Denken bringen, Sprendlingen (Kap.12)

TROST,H. (1984) : Wissenspräsentation in der AI am Beispiel Semantischer Netze, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 47 ff.

Kap. 9 : Suchprogramm

GRAHAM,N. (1983) : Künstliche Intelligenz, wie Sie Ihren Computer zum Denken bringen, Sprendlingen (Kap. 3 bis 5)

HARTNELL,T. (1984) : Exploring Artificial Intelligence on your Microcomputer, London / Melbourne (Sektion 2)

HORN,W. (1984) : Methoden der Artificial Intelligence, in : Arificial Intelligence, eine Einführung (Hg. : L.Richter / W.Stucky), Stuttgart, Seite 27 ff.

VOSS,W. (1972) : Minimierungsaufgaben in der Statistik, Köln

Kap. 10 : Entscheiden

BIBEL,W. (1984) : Automatische Inferenz, in : Artificial Intelligence, eine Einführung (Hg. : L.Richter / W.Stucky), Stuttgart, Seite 145 ff.

TIEDE,M. / VOSS,W. (1982) : Induktive Statistik, Teil 1, Köln

TIEDE,M. / VOSS,W. (1982) : Induktive Statistik, Teil 2, Köln

Kap.11 : Erkennen

BRAIN,K. / BRAIN,S. (1984) : Artificial Intelligence on the commodore 64, make your micro think, Exeter (Kap. 3 und 8)

BUCHBERGER,E. (1984) : Sprachverstehen in der Artificial Intelligence, in : Artificial Intelligence, eine Einführung (Hg. L.Richter / W.Stucky), Stuttgart, Seite 125 ff.

HORN,W. (1984) : Methoden der Artificial Intelligence, in : Artificial Intelligence, eine Einführung (Hg.: L.Richter / W.Stucky), Stuttgart, Seite 27 ff.

HORX,M. (1984) : Chip Generation, Reinbek

STEDE,M. u.a. (1984) : Einführung in die künstliche Intelligenz, Band 2 : Anwendungsgebiete, Sprendlingen (Kap.3)

Kap.12 : Taktisches Spiel

GRAHAM,N. (1983) : Künstliche Intelligenz, wie Sie Ihren Computer zum Denken bringen, Sprendlingen (Kap.6 und 7)

MENGES,G. (1969) : Grundmodelle wirtschaftlicher Entscheidungen, Einführung in moderne Entscheidungstheorien, Köln / Opladen

WALKOWIAK (1984) : Adventures und wie man sie programmiert, Düsseldorf

Kap.13 : Selbstlernendes Programm

BRAIN,K. / BRAIN,S. (1984) : Artificial intelligence on the commodore 64, make your micro think, Exeter (Kap. 6)

HARTNELL,T. (1984) : Exploring Artificial Intelligence on your Microcomputer, London / Melbourne (Kap. 2 und 13)

RETTI,J. (1984) : Knowledge Engineering und Expertensysteme, in : Artificial Intelligence, eine Einführung (Hg. : L.Richter / W.Stucky), Stuttgart, Seite 73 ff.

Kap.14 : Kunst

HARTNELL,T. (1984) : Exploring Artificial Intelligence on your Microcomputer, London / Melbourne (Kap. 10)

Kap.15 : Dialog

BRAIN,K. / BRAIN,S. (1984) : Artificial Intelligence on the commodore 64, make your micro think, Exeter (Kap. 4 und 5)

BUCHBERGER,E. (1984) : Sprachverstehen in der Artificial Intelligence, in : Artificial Intelligence, eine Einführung (Hg. : L.Richter / W.Stucky), Stuttgart, Seite 125 ff.

GRAHAM,N. (1983) : Künstliche Intelligenz, wie Sie Ihren Computer zum Denken bringen, Sprendlingen (Kap.13)

HARTNELL,T. (1984) : Exploring Artificial Intelligence on your Microcomputer, London / Melbourne (Sektion 3)

Kap.16 : Reaktion

GRAHAM,N. (1983) : Künstliche Intelligenz, wie Sie Ihren Computer zum Denken bringen, Sprendlingen (Kap.9)

STEDE,M. u.a. (1984) : Einführung in die künstliche Intelligenz, Band 2 : Anwendungsgebiete, Sprendlingen (Kap.4)

Kap.17 : Unterricht

BRAIN,L. / BRAIN,S. (1984) : Artificial intelligence on the commodore 64, make your micro think, Exeter (Kap.9)

HARTNELL,T. (1984) : Exploring Artificial Intelligence on your Microcomputer, London / Melbourne (Kap.9)

STICHWORTVERZEICHNIS

** A **

ABAKUS 56
ABBRUCHKRITERIUM 199
ABFRAGE 106
ACTION-SPIEL 354
ADRESSE 111
ADVENTURE 254
AI 74
ANWEISUNG 90
ARBEIT, INTELLIGENTE 2
ARITHMETIK 56
ARITHMETISCHES MITTEL 120
ASCII-CODE 92, 93
AUSGABEGERÄT 95
AUSKUNFTSPROGRAMM 287
AUSKUNFTSSYSTEM 173, 288

** B **

BABBAGE 58
BASIC 97
BEFUND 78
BEGABUNG 11
BETRIEBSSYSTEM 90
BEWUßTSEIN 22, 24, 37
BILDSCHIRM 95
BILDSCHIRMPOSITION 111
BILDSCHIRMSPEICHERADRESSE 262, 316
BILDSCHIRMZONE 99
BINOMIALVERTEILUNG 219, 222
BINÄRES SUCHEN 195
BINÄRPRINZIP 91
BINÄRSYSTEM 91
BIT 92
BRETTSPIEL 54
BYTE 93

** C **

C64, COMMODORE 4
CHEMISCHES ELEMENT 177
CHIP 63
CHR$ 102, 109
CODE 92
COMPUTER 56, 68, 85
COMPUTERSPIEL 65
CPU 96

** D **

DATA 104
DATEN 88, 89
DATENBANK 73
DATENVERARBEITUNG 63, 88, 90
DEMONSTRATIONSBEISPIEL 132
DENKEN 38
DENKENDE MASCHINE 55
DENKFÄHIGKEIT 27
DEZIMALSYSTEM 57, 91
DIALOG 325
DIALOGGERÄT 95
DIFFERENTIALRECHNUNG 193
DISKETTE 95
DRUCKER 95

** E **

ELEKTRONENGEHIRN 61, 62
ELEKTRONENRECHNER 61
ELEKTRONENRÖHRE 61
ELEKTROTECHNIK 59
ELEMENT, CHEMISCHES 177
ELIZA 326
EMPIRISCHE FORSCHUNG 213
EMPIRISCHE WISSENSCHAFT 78
END 99

ENGINEERING, KNOWLEDGE 309
ENIAC 61
ENTSCHEIDEN 213
ENTSCHEIDUNG 25, 77, 213
ENTSCHEIDUNGSBAUM 256
ERFAHRUNG 14
ERGEBNISAUSGABE 98
ERKENNEN 80, 237
ERKENNTNISGEWINNUNG 18
EVA-PRINZIP 120
EXPERTENSYSTEM 73, 137, 174, 287, 288
EXTERNER SPEICHER 95
EXTREMWERT 193

** F **

FAKULTÄT 219
FARBADRESSE 112
FARBE 112, 113
FARBSPEICHERADRESSE 262
FEHLENTSCHEIDUNG 87
FELD 89, 101
FLUßDIAGRAMM 118
FOR...TO 108
FORSCHUNG, EMPIRISCHE 213
FUNKTION 102, 109
FUNKTIONSWEISE 87

** G **

GEDÄCHTNIS 41, 42
GEISTIGE TÄTIGKEIT 2, 28
GENERATION, FÜNFTE 2, 47
GET 103
GOSUB 106
GOTO 105
GRAPHIK 110
GRAPHIKSYMBOL 110
GRUNDPRINZIP 5

** H **

HAUPTPROGRAMM 106
HEXADEZIMALSYSTEM 92
HEXADEZIMALZAHL 93
HINTERGRUND, THEORETISCHER 7
HOLLERITH 59
HOLLERITH-MASCHINE 60
HOME-COMPUTER 46, 64, 88
HYPOTHESE 78, 213

** I **

IF 106
ILLUSTRATIONSBEISPIEL 5
INDIZIERTE VARIABLE 104
INDUSTRIEROBOTER 16, 19
INFORMATION 25, 89
INFORMATIONSBASIS 175
INFORMATIONSEINGABE 103
INFORMATIONSEINHEIT 91, 92
INFORMATIONSTHEORIE 25
INFORMATIONSÜBERTRAGUNG 59
INPUT 103
INT 102
INTELLIGENCE, ARTIFICIAL 74
INTELLIGENTE ARBEIT 2
INTELLIGENZ 10, 11
INTELLIGENZ, KÜNSTLICHE 46, 67

** J **

JACQUARD 58

** K **

KASSETTE 95
KOHLENSTOFF 39
KOMMANDO 90, 100
KOMMUNIKATIONSTECHNOLOGIE 65
KUNST 83, 311
KÜNSTLICHE INTELLIGENZ 46, 67

** L **

LABYRINTH 288
LADEN 95
LAUFVARIABLE 108
LEBEN 18
LEBENSBEWÄLTIGUNG 11, 12
LEBEWESEN, SOZIALES 22
LEFT$ 109
LEHRER 355
LEIBNIZ 58
LEN 109
LERNEN 41
LERNFÄHIGKEIT 14
LERNVERMÖGEN 14
LET 101
LIST 100
LOCHKARTE 59
LOGIK 55
LOGISCHER SCHLUß 55
LULLUS 55
LYRIK 312

** M **

MALEREI 312
MASCHINE 68
MASCHINE, DENKENDE 55
MATERIE 20
MECHANISCHER RECHNER 57
MENÜ 188
MENÜ-PROGRAMMIERUNG 188
MID$ 109
MIKROPROZESSOR 63
MIKROPROZESSORTECHNIK 63
MITTEL, ARITHMETISCHES 120
MODEM 238
MODUL 63
MODUL-PROGRAMMIERUNG 261
MUSIK 312
MUSTERERKENNUNG 81, 237

** N **

NAME 89, 101
NEW 100
NEXT 108
NORMALGRAPHIK 111
NORMALVERTEILUNG 222
NULLSTELLE 197

** O **

ON...GOTO 107, 188
OPERATOR 101
ORDNUNG 18
ORDNUNGSGEWINN 25

** P **

PASCAL 58
PERFORMANZMODUS 29
PERIODENSYSTEM 177
PFAD 77
PFLANZE 19
POKE 111
PRINT 98
PROBLEM 14, 23
PROBLEMANALYSE 117, 119
PROGRAMM 1, 58, 90
PROGRAMM, SELBSTLERNENDES 82, 287
PROGRAMMABLAUFPLAN 118
PROGRAMMIERSPRACHE 90
PROGRAMMIERUNG 58
PROZEßSTEUERUNG 58, 65

** R **

READ 104
REAKTION 341
RECHENGESCHWINDIGKEIT 59
RECHNER 56
RECHNER, MECHANISCHER 57
RECHNERARCHITEKTUR 64
RELAIS 60
REM 100
RETURN 107
RIGHT$ 109
RND 102
ROBOTIK 132, 341
RUN 99

** S **

SATZ 98
SCHACHPROGRAMM 81
SCHACHSPIEL 55
SCHICKARD 57
SCHLEIFE 107
SCHLUß, LOGISCHER 55
SELBSTBEWUßTSEIN 22
SELBSTLERNENDES PROGRAMM 82, 287
SIGNIFIKANZNIVEAU 218
SIGNIFIKANZTEST 216
SILIZIUM 39
SIMULATIONSMASCHINE 72
SIMULATIONSMODUS 29
SOFTWARE 65
SORTIEREN 128
SOZIALES LEBEWESEN 22
SPEICHERFELD 89
SPEICHERKAPAZITÄT 93
SPEICHERN 95
SPEICHERPLATZ 89
SPEICHERTECHNOLOGIE 2
SPIEL 81
SPIEL, TAKTISCHES 253
SPIELPROGRAMM 191
SPIELTHEORIE 257
SPRACHE 43
SPRACHERKENNUNG 43
SPRACHVERSTEHEN 81
SPRUNG 105
SOR 102

STATEMENT 98
STICHPROBE 217
STR$ 110
STRING 89, 109
STRINGBEARBEITUNG 109
STUFENZAHL 91
SUCHEN 191
SUCHEN, BINÄRES 195
SUCHKOSTEN 192
SUCHPFAD 191
SUCHPROGRAMM 174, 191
SUCHSYSTEM 76, 287
SUCHZEIT 192
SYSTEM 20

** T **

TAB 99
TABLETT 238
TAKTISCHES SPIEL 253
TASTATUR 95, 261
TEST 216
TESTVERFAHREN 235
TEXTVERARBEITUNG 65
THEORETISCHER HINTERGRUND 7
THEORIE 72
TIER 19
TRANSISTOR 62
TURING-TEST 48
TYP 101
TÄTIGKEIT, GEISTIGE 2, 28

** U **

UEBERSCHREITUNGSWAHRSCHEINLICHKEIT 221
UEBERSETZUNGSPROGRAMM 92, 174
UNTERPROGRAMM 106, 261
UNTERRICHT 355

** V **

VARIABLE 89, 101
VARIABLE, INDIZIERTE 104
VERGANGENHEIT 53
VERHALTEN, ZIELGERICHTETES 12
VERNETZUNG 66
VIRUS 20

** W **

WAHRSCHEINLICHKEIT 215
WAHRSCHEINLICHKEITSSTATISTIK 215
WENDEPUNKT 193
WERKZEUG 27, 54
WERT 89
WERTZUWEISUNG 101
WISSENSCHAFT, EMPIRISCHE 78
WISSENSERWERB 80
WÜRFELSPIEL 285

** Z **

Z1 60
ZAHL 89
ZAHNRAD 57
ZEICHENKETTE 89
ZEILENVORSCHUB 99
ZENTRALEINHEIT 95, 96
ZIELFUNKTION 192, 193
ZIELGERICHTETES VERHALTEN 12
ZUFALL 216
ZUSE 60
ZÄHLIMPULS 56
ZÄHLIMPULSÜBERTRAGUNGSELEMENT 56
ZÄHLPROZEß 56

DATA BECKER'S NEUE BÜCHER UND PROGRAMME FÜR COMMODORE

Spickzettel ade.

Ein neues DATA BECKER BUCH, das den Einsatz des COMMODORE 64 in der Schule entscheidend mitprägen dürfte, wurde von Professor Voß geschrieben. Besonders für Schüler der Mittel- und Oberstufe geschrieben, enthält das Buch viele interessante Problemlösungs- und Lernprogramme, die besonders ausführlich und leicht verständlich beschrieben sind. Sie ermöglichen ein intensives und anregendes Lernen, unter anderem mit folgenden Themen: Satz des Pythagoras, quadratische Gleichungen, geometrische Reihen, Pendelbewegungen, mechanische Hebel, Molekülbildung, exponentielles Wachstum, Vokabeln lernen, unregelmäßige Verben, Zinseszinsrechnung. Ein kurzer Überblick über die Grundlagen der EDV, eine knappe Wiederholung der wichtigsten BASIC-Elemente und eine Einführung in die Grundzüge der Problemanalyse vervollständigen das Ganze. Mit diesem Buch machen die Hausaufgaben wieder Spaß!

DAS SCHULBUCH ZUM COMMODORE 64, 1984, über 300 Seiten, DM 49,–

Tempo!

MASCHINENSPRACHE FÜR FORTGESCHRITTENE ist bereits das zweite Buch von Lothar Englisch zum Thema Maschinenprogrammierung mit dem COMMODORE 64. Hier wird von der Problemanalyse bis zum Maschinensprachealgorithmus in die Grundlagen der professionellen Maschinensprache-programmierung eingeführt. In diesem Buch finden Sie unter anderem folgende Themen behandelt: Problemlösungen in Maschinensprache, Programmierung von Interruptroutinen, Interruptquellen beim COMMODORE 64, Interrupts durch CIA's und Videocontroller, Programmierung der Ein-Ausgabe-Bausteine, die CIA's des COMMODORE 64, Timer, Echtzeituhr, parallele und serielle Ein/Ausgabe, BASIC-Erweiterungen, Programmierung eigener BASIC-Befehle und -Funktionen, Möglichkeiten zur Einbindung ins Betriebssystem sowie viele weitere Tips & Tricks zur Maschinenprogrammierung. Dieses Buch sollte jeder haben, der wirklich intensiv mit der Maschinensprache des COMMODORE 64 arbeiten will.

MASCHINENSPRACHE FÜR FORTGESCHRITTENE, 1984, ca. 200 Seiten, DM 39,–

Macht Druck.

DAS GROSSE DRUCKERBUCH für Drucker-Anwender mit COMMODORE-Computern ist endlich da! Es enthält eine riesige Sammlung von Tips & Tricks, Programmlistings und Hardwareinformationen. Rolf Brückmann und Klaus Gerits beschäftigen sich mit Sekundäradressen, Anschluß einer Schreibmaschine am Userport, Druckerschnittstellen (Centronics, V 24, IEC-Bus), hochauflösender Grafik, Text- und Grafikhardcopy, Grafik mit Standardzeichensatz, formatierter Datenausgabe, Plakatschrift, Textverarbeitung und vieles mehr. Zusätzlich wird das Betriebssystem des MPS801 zerlegt, mit Prozessorbeschreibung (8035), Blockschaltbild und einem ausführlich kommentierten ROM-Listing. Thomas Wiens schrieb den Teil über die Programmierung des Plotters VC-1520: Handhabung des Plotters, Programmierung von Sonderzeichen, Funktionendarstellung, Kuchen und Säulendiagramme, Entwurf dreidimensionaler Gegenstände. Natürlich wieder viele interessante Listings. Unentbehrlich für jeden, der einen COMMODORE 64 oder VC-20 und einen Drucker besitzt.

DAS GROSSE DRUCKERBUCH, 1984, über 300 Seiten, DM 49,–

Tausend-sassa.

Fast alles, was man mit dem COMMODORE 64 machen kann, ist in diesem Buch ausführlich beschrieben. Es ist nicht nur spannend zu lesen wie ein Roman, sondern enthält neben nützlichen Programmlistings vor allem viele, viele Anwendungsmöglichkeiten des C64. Dabei wurde besonderer Wert darauf gelegt, daß das Buch auch für Laien leicht verständlich ist. Eine Auswahl aus der Themenvielfalt: Gedichte vom Computer, Einladung zur Party, Diplomarbeit – professionell gestaltet, individuelle Werbebriefe, Autokosten im Griff, Baukostenberechnung, Taschenrechner, Rezeptkartei, Lagerliste, persönliches Gesundheitsarchiv, Diätplan elektronisch, intelligentes Wörterbuch, kleine Notenschule, CAD für Handarbeit, Routenoptimierung, Schaufensterwerbung, Strategiespiele. Teilweise sind Programmlistings fertig zum Eintippen enthalten, soweit sich die „Rezepte" auf 1–2 Seiten realisieren ließen. Wenn Sie bisher nicht immer wußten, was Sie mit Ihrem 64er alles anfangen sollten, nach dem Lesen des IDEENBUCHES wissen Sie's bestimmt!

DAS IDEENBUCH ZUM COMMODORE 64, 1984, über 200 Seiten, DM 29,–